НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 6.050102 «Комп'ютерна інженерія»

на тему: «Система моніторингу для розумного дому (клієнтська частина)»

Виконав:
студент IV курсу, групи ІО-61
Добровольський Дмитро Дмитрович _____

Керівник:
проф. д. т. н.
Сімоненко Валерій Павлович _____

Консультант нормоконтроль:
проф. д. т. н.
Сімоненко Валерій Павлович _____

Рецензент:
доц. каф. СКС
Орлова Марія Миколаївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 6.050102 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ СЕРГІЙ СТІРЕНКО
«___»_____20__ р.

ЗАВДАННЯ
на дипломний проєкт студенту
Добровольському Дмитру Дмитровичу

1. Тема проєкту «Система моніторингу для розумного дому (клієнтська частина)», керівник проєкту Сімоненко Валерій Павлович, проф., д.т.н., затверджені наказом по університету від «07» травня 2020 р. №1081-С

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту *Технічна документація. Теоретичні та статистичні дані. Система на кристалі ESP8266 та плата ESP8266. Датчик температури та вологості DHT21. Дисплей OLED SSD1306 Середовище розробки PyCharm, MicroPython.*

4. Зміст пояснювальної записки *Аналіз предметної області, дослідження методики та засобів створення пристроїв на базі плати NodeMCU v3, розробка пристрою для збору показників температури та вологості.*

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо) блок-схема алгоритму, функціональна схема, структурна схема.

6. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	*Затвердження теми роботи*	*01.09.2019*	
2	*Вивчення та аналіз завдання*	*30.03.2020*	
3	*Вибір апаратних та програмних засобів*	*10.04.2020*	
4	*Розробка апаратної частини пристрою*	*22.04.2020*	
5	*Програмна реалізація системи*	*04.05.2020*	
6	*Оформлення пояснювальної записки*	*17.05.2020*	
7	*Передзахист*	*26.05.2020*	
8	*Захист*	*16.06.2020*	

Студент Дмитро ДОБРОВОЛЬСЬКИЙ

Керівник Валерій СІМОНЕНКО

Анотація

Бакалаврська дипломна робота присвячена розробці клієнтської частини системи моніторингу вимірювання температури та вологості для розумного дому. Тема роботи є особливо актуальною у сьогодення, враховуючи швидкі темпи росту концепції «Інтернету речей» у галузі інформаційних технологій. У роботі проводиться огляд існуючих рішень в даній області; порівняння та вибір комплектуючих для розробки власного рішення враховуючи економічний та естетичний фактори; розробка програмного забезпечення для робочого прототипу. Результатом роботи є готовий пристрій та супутнє програмне забезпечення для практичного використання.

Abstract

The bachelor's thesis is devoted to the development of the client part of the monitoring system for measuring temperature and humidity for a smart home. The topic of the work is especially relevant today, due to the rapid growth of the concept of "Internet of Things" in the field of information technology. The graduation work reviews the existing solutions in this area; comparison and selection of components to develop your own solution taking into account economic and aesthetic factors; software development for a working prototype. The result is a ready-made device and related software for practical use.

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломної роботи
освітньо-кваліфікаційного рівня бакалавр

на тему: “Система моніторингу для розумного дому (клієнтська частина)”

Київ – 2020 року

Технічне завдання до *дипломної* роботи

Зміст

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: «Пристрій для вимірювання температури та вологості навколишнього середовища».

Область застосування: альтеративна розробка існуючим системам моніторингу навколишнього середовища для розумного дому.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки служить завдання на виконання розробки системи моніторингу для розумного дому, затвердженого кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний Інститут імені Ігоря Сікорського»

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою данної роботи є розробка апаратної частини системи моніторингу для розумного дому.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами для розробки служать науково-технічна література з комп`ютерних технологій, довідники з електроніки, публікації в періодичних виданнях та Інтернеті за даним питанням.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розроблюваного продукту

- Вимірювання температури навколишнього середовища;
- Вимірювання вологості навколишнього середовища;
- Відображення виміряних показань, а також подальша передача їх на сервер;

5.2. Вимоги до програмного забезпечення

- Операційна система Linux, Windows, MacOS або інші UNIX-подібні системи;
- Доступ до мережі Інтернет;

5.3. Вимоги до апаратного забезпечення

- Джерело живлення 5 В, 1-2 А;
- Наявність Wi-Fi точки доступу;

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення літератури	30.01.2020
Складання і узгодження технічного завдання	11.02.2020
Створення тестового стенду та робочого прототипу	14.03.2020
Створення програмного забезпечення	10.04.2020
Тестування системи в цілому та її налаштування	04.05.2020
Оформлення документації дипломної роботи	17.05.202

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

до дипломної роботи
освітньо-кваліфікаційного рівня бакалавр

на тему: “Система моніторингу для розумного дому (клієнтська частина)”

Київ – 2020 року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	ДП.467100.001 ТЗ	Технічне завдання	4	
3	А4	ДП.467100.002 ВП	Відомість проєкту	1	
4	А4	ДП.467100.003 ПЗ	Пояснювальна записка	58	
5	А4	ДП.467100.004 Д1	Схема структурна	1	
6	А4	ДП.467100.005 Д2	Блок-схема алгоритму	1	
7	А4	ДП.467100.006 Д3	Схема функціональна	1	

					ІАЛЦ.467100.002 ВП	
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розроб.		*Добровольський Д. Д.*			*Відомість дипломного проекту*	*Літ.* / *Арк.* 11 / *Акрушів* 1
Перевір.		*Сімоненко В.П.*				
Н. Контр.		*Сімоненко В.П.*				*НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІО-61*
Затверд.						

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту

на тему: “Система моніторингу для розумного дому (клієнтська частина)”

Київ – 2020 року

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

UART – universal asynchronous receiver-transmitter

GPIO – general-purpose input/ouput

RFID – Radio Frequency IDentification

SoC – System-on-a-Chip

OLED – organic light-emitting diode

HTTP – HyperText Transfer Protocol

I2C – Inter-Integrated Circuit

REPL – read-eval-print loop

SDK – software development kit

ВСТУП

У сьогодення можна прослідкувати проникнення цифрових технологій майже у всі існуючі галузі та повсякденне життя людства, йде все більша інтеграція між різними сферами діяльності, що в свою чергу допомагає відкрити нові можливості для розвитку окремим компаніям та цілим конгломератам в певних галузях. Така тенденція спричинена багатьма факторами, одним з найбільших є велике поширення та розвиток мережевих технологій та засобів зв`язку. З плином часу, Інтернет ставав все більш доступною та всеохоплюючою технологією в світі, а кількість пристроїв, що використовують інтернет-послуги збільшується з кожним днем. Це призводить до виникнення величезного потоку даних та інформації, що надходить з різних джерел, збір та аналіз яких з використанням сучасних засобів обробки в свою чергу допомагає компаніям точніше передбачати розвиток у тих чи інших течіях діяльності та знаходити нові рішення. Разом із цим, збільшується кількість напрямів, які потрібно контролювати та керувати, а також необхідність швидкого реагування на постійно змінні показники різних систем задля отримання максимальної вигоди від конкретної ситуації, що призводить до необхідності автоматизації великої кількості процесів в сучасних реаліях.

Акумулювання всіх вище представлених вимог найкраще викладено в так званій концепції «Інтернету речей», суть якої полягає у поєднанні різної кількості «розумних» цифрових пристроїв, спільних за призначенням в єдину мережу. Дана технологія може використовуватись в широкому спектрі місць застосування від великих та складних технологічних процесів виробництва до локальних та домашніх систем моніторингу та автоматизації. Бурхливий розвиток «Інтернету речей» був спричинений як стрімким ростом бездротових мережевих технологій, так і революції в обчислювальній техніці.

Дійсно, потужність сучасних процесорів за останні десятиліття збільшилась в десятки та сотні разів, а мікроконтролери отримали велике розповсюдження та доступність завдяки низькій ціні. В сучасному світі все більш стає популярним один із варіантів «Інтернету речей» у вигляді «розумного» дому – автоматизованої системи моніторингу та контролю за певними параметрами навколишнього середовища (температури, вологості, освітлення та ін.), а також управління за певними щоденними задачами.

РОЗДІЛ 1

ОГЛЯД ТЕХНОЛОГІЇ «ІНТЕРНЕТ РЕЧЕЙ»

Інтернет речей (IoT) лише нещодавно увійшов у наше повсякденне життя. Я вважаю, що на теперішній час ця технологія оточує нас скрізь, де б ми не знаходились: пристрої домашньої автоматики, розташовані в будинку; старт-датчики в офісах, вбудовані в робочі місця; фітнес-трекери та розумні годинники, що ми носимо. Як повідомляє онлайн портал Statista, лише вони створюють масову екосистему з 26,66 мільярдів взаємопов'язаних пристроїв, які мають неабиякий вплив на суспільства та економіку у всьому світі. Але світ не завжди був таким, як ми його бачимо зараз. До 1999 року термін "Інтернет речей" ще не існував. Сама концепція підключених пристроїв в певну мережу датується 1832 р., коли був спроектований перший електромагнітний телеграф. Телеграф забезпечив прямий зв'язок між двома машинами через передачу електричних сигналів. Однак справжня історія IoT почалася з винайдення Інтернету в кінці 1960-х, який потім швидко розвивався протягом наступних десятиліть. Першим підключеним пристроєм був торговий автомат Coca-Cola, розташований в Університеті Карнегі-Дині, яким керували місцеві програмісти. Вони інтегрували мікроперемикачі в машину і використовували ранню форму Інтернету, щоб перевірити, чи охолоджуючий прилад підтримує напої досить холодними та чи є в наявності банки з напоями. Цей винахід сприяв подальшим дослідженням у цій галузі та розвитку взаємопов'язаних машин у всьому світі. У 1990 році Джон Ромкі вперше підключив тостер до Інтернету за допомогою протоколу TCP/IP. Через рік вчені Кембриджського університету виступили з ідеєю використати перший прототип веб-камери для контролю кількості кави, наявної в кавовому апараті комп'ютерної лабораторії. Вони запрограмували веб-камеру три рази в хвилину фотографувати кавовий горщик, а потім надсилати зображення на комп’ютери

в локальній мережі, таким чином дозволяючи всім бачити, чи наявна кава в кавовій машині.

1999 рік є одним з найбільш значущих для історії IoT, адже саме тоді Кевін Ештон вперше ввів термін "Інтернет речей". Технолог Ештон, виступав з презентацією для Procter & Gamble, де описав IoT як технологію, яка з'єднала кілька пристроїв за допомогою тегів RFID для управління поставками в сфері менеджменту. Він спеціально використовував слово "Інтернет" у назві своєї презентації, щоб привернути увагу аудиторії, оскільки Інтернет на той час саме починав свій бурхливий розвиток. Незважаючи на те, що його уявлення про підключення пристроїв на основі RFID відрізняється від сьогоднішнього IoT напрямку, прорив Ештона відіграв важливу роль в історії інтернету речей та технологічного розвитку в цілому.

На початку 21 століття термін "Інтернет речей" набув широкого вжитку в засобах масової інформації, тоді його широко почали вживати представництва "Гардіан", "Форбс" та "Бостонський глобус". Інтерес до технології IoT постійно збільшувався, що призвело до 1-ї Міжнародної конференції з Інтернету речей, що відбулася у Швейцарії у 2008 році, де учасники 23 країн обговорили RFID, бездротовий зв'язок короткого діапазону та сенсорні мережі. Крім того, деякі розробки всесвітньо відомих технічних компаній сприяли еволюції IoT. Одним з таких винаходів був холодильником, підключений до Інтернету, який був представлений LG Electronics у 2000 році, що дозволяв його користувачам робити покупки в Інтернеті та здійснювати відеодзвінки. Ще однією істотною розробкою став маленький робот у формі кролика на ім’я Набазтаг, створений у 2005 році, який міг розповісти останні новини, прогноз погоди та зміни на фондовому ринку. За даними Cisco, вже тоді кількість взаємопов'язаних пристроїв перевершувало кількість людей на Землі. На рис. 1 зображено еволюцію інтернету речей та прогнозовану кількість пристроїв, що входять до мережі в наступних роках.

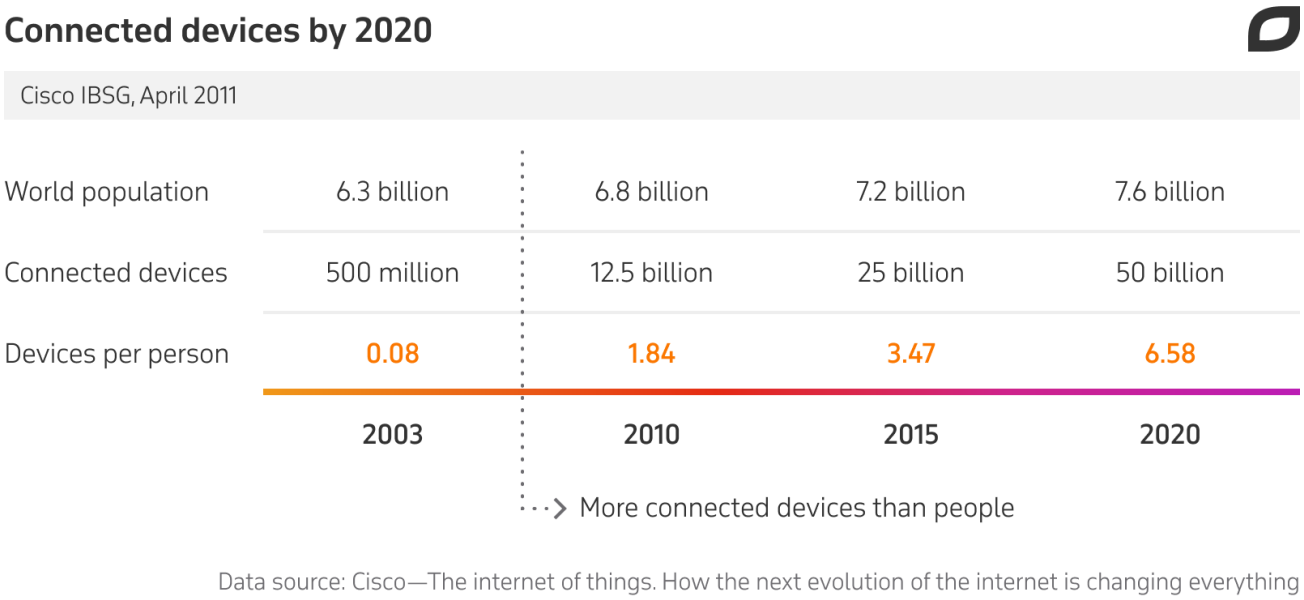


Рис. 1.1. Еволюція IoT

Бум IoT був підтриманий його доповненням до циклу Hartner Hype для нових технологій у 2011 році. У тому ж році IPv6 був відкритий протокол мережевого рівня, який є цільовим для розвитку IoT. З того часу концепція взаємопов'язаних між собою пристроїв стала широко розповсюдженою та звичною в нашому повсякденному житті. Глобальні технологічні гіганти, такі як Apple, Samsung, Google, Cisco та General Motors, зосереджують свої зусилля на виробництві датчиків і пристроїв IoT - від взаємопов’язаних термодатчиків та розумних окулярів до автоцентрів, що керують авто. IoT знайшов свій шлях майже до кожної галузі: виробництво, охорона здоров'я, транспорт, нафта та енергетика, сільське господарство, роздрібна торгівля та багато інших. Цей драматичний зсув переконав нас, що революція IoT відбувається саме зараз. На сьогоднішній день платформ IoT зберігають міцну позицію серед провідних тенденцій цьогорічного циклу Gartner Hype завдяки розвитку віртуальних помічників та голосових асистентів, взаємопов’язаних будівель та автомобілями, що керуються штучним інтелектом. Технологія досягне свого плато продуктивності за 5–10 років.

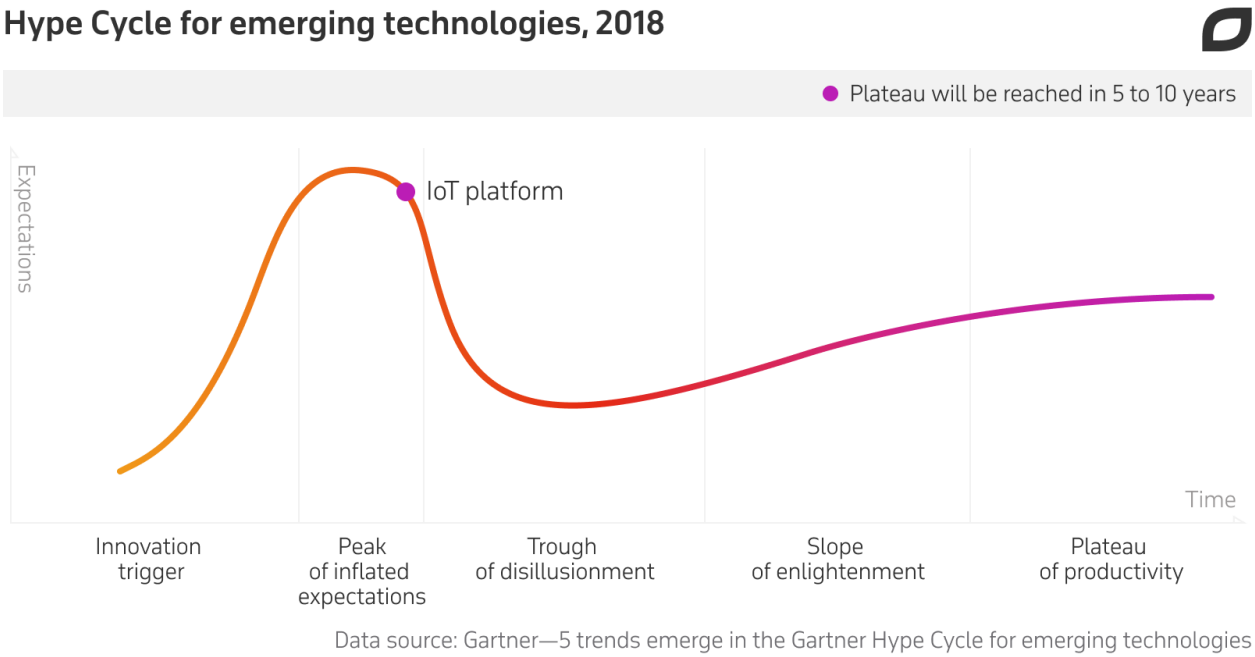


Рис. 1.2. Стан IoT технології в циклі Gartner

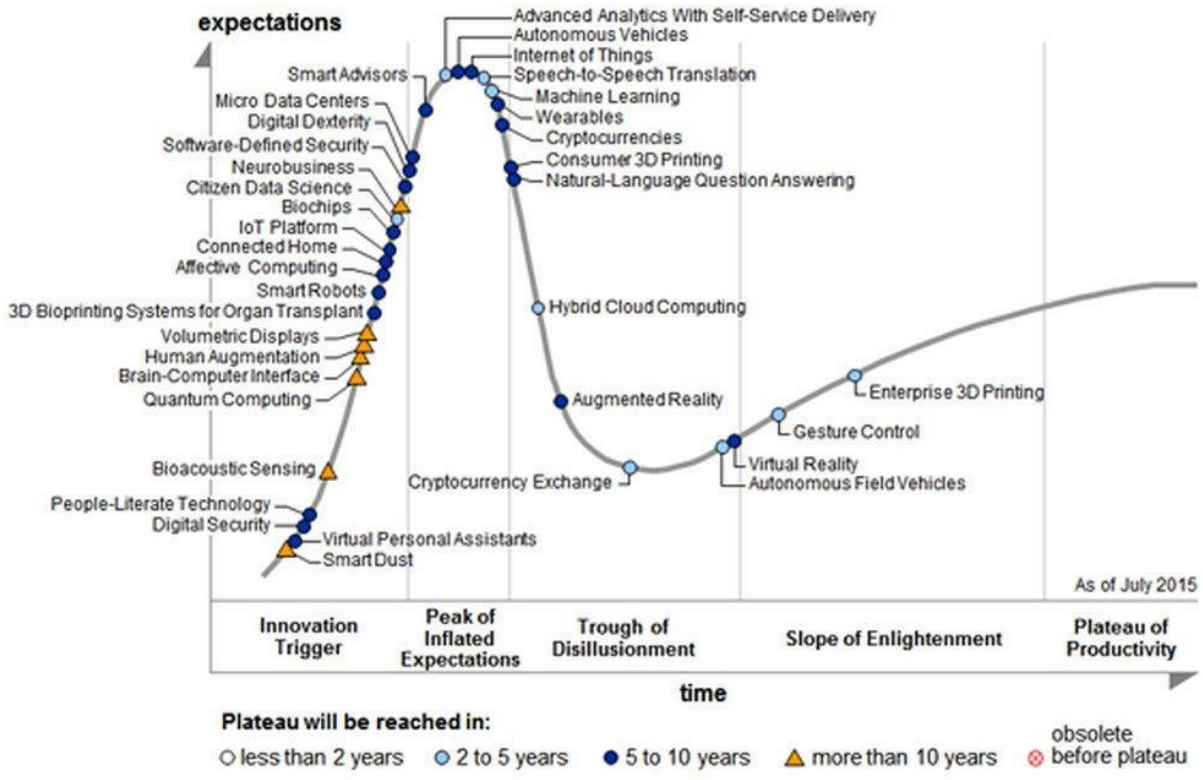


Рис. 1.3. Стан IoT технології в циклі Gartner в порівнянні з іншими технологіями

1.1 Статус технології на поточний час та її майбутнє

Враховуючи стрімкий темп розвитку, IoT незабаром домінує у світі. У 2019 році компанія Gartner передбачила, що корпоративний і автомобільний IoT ринок зросте до 5,8 мільярда взаємопов’язаних точок у 2020 році, що означатиме збільшення на 21% порівняно з 2019. Все, що можна підключити, буде підключено, тим самим утворюючи комплексну цифрову систему, в якій всі пристрої спілкуються з людьми і один з одним. Ось кілька найважливіших факторів, що сприяють цьому швидкому розширенню IoT:

- Зниження ціни апаратного забезпечення (датчиків та сенсорів, мікроконтролерних платформ)
- Зниження витрат на збір та зберігання даних завдяки хмарним рішенням
- Широке розширення підключення до Інтернету
- Збільшення обчислювальної потужності
- Зростання кількості смартфонів та планшетів

Безперечно, швидкий ріст IoT кардинально змінить світ, у якому ми живемо. Наше неминуче взаємопов'язане майбутнє, безумовно, принесе багато цінності та захоплюючих можливостей для людей.

1.2 Концепція «Розумного» будинку

Багато визначень системи «розумний будинок» підкреслюють, що їїконцепція полягає у підключенні датчиків, приладів та пристроїв через мережу зв'язку з метою віддаленого моніторингу, доступу та контролю житлового середовища. Більше того, важливим моментом системи є надання послуг, що відповідають потребам користувачів. Лише останнім часом термін "розумний" або "смарт" пов'язаний з енергоефективністю будівель. Таким ном, її часто називають системою управління енергоресурсами дому, яка керує

споживанням енергії всередині будівель. Зокрема, ці системи спрямовані на максимальне та оптимальне використання електроенергії в будинках контроль домашнього обладнання для зменшення та оптимізації використання енергії, а також створення та накопичення енергії з різних джерел. Насправді найважливішою технологією в системі розумних будівель є сама мережа, що дозволяє обмін інформацією в реальному часі від будівель та користувачів до інших клієнтських будівель та користувачів шляхом підключення та узгодження всіх технологічні пристроїв, встановлені в системі по певним протоколам зв'язку, а також наявність центрального вузла, що відповідає за цю мережу та контроль за всіма ресурсами.

Хоча може бути очевидним, що сучасні будинки з високим обслуговуванням потребують вдосконаленої системи управління, вона повинна слід усвідомити, що простіші будівлі, що покладаються на опалювальний котел та природну вентиляцію, все ще можуть отримати користь від сучасних система управління енергією. Зростаючий акцент на економії енергії та скороченні парникових газів викиди служать для підвищення важливості ефективного контролю. Однак такий підхід - не єдиний спосіб зменшити споживання. Краще, ефективніше використання традиційних приладів та компонентів може бути однаково ефективно.

Тому, поряд із просуванням енергоефективного обладнання, розробники політики з енергоефективності вносять у маси ще однин спосіб зменшити споживання, тим самим підвищивши ефективність системи. Насправді підвищення ефективності системи – єдине підхід до багатьох існуючих будівель, які з культурних та історичних причин не можуть бути модернізовані сучасні технології будівництва. Підвищення ефективності системи вимагає централізованого централізованого контролю над усією енергією пов'язані компоненти, щоб гарантувати, що вся система може працювати з максимальною ефективністю. Причому спілкування мережа має вирішальне значення для того, щоб збирати дані з усіх компонентів Smart Home та

детально розробляти їх відповідні стратегії збільшення економії енергії. На ефективність також впливає здатність системи до повідомляти користувачів про їх використання енергії та змінювати їх звички, тому спосіб спілкування використовується для їх інформування (за допомогою ІКТ чи інших методів) є основоположним. Постійний моніторинг компонентів ОВК чітко враховується бути корисним у зменшенні загального споживання енергії в будинках. Значна економія електроенергії - від 5% до в середньому 15% можна досягти, зрозумівши деталі використання енергії на рівні окремих пристроїв всередині будинків , надаючи інформацію користувачам про зв’язок між енергією та виконаною дії на пристроях. [16]

1.3. Огляд існуючих технологій «розумного дому»

Розумний дім має три компоненти: апаратне, програмне забезпечення та протоколи зв'язку. Він має широкий спектр застосувань для цифрового споживача. Деякі з областей домашньої автоматизації керували підключенням IoT, наприклад: контроль освітлення, садівництво, безпека та безпека, якість повітря, моніторинг якості води, голосові помічники, вимикачі, замки, лічильники енергії та води.

До вдосконалених компонентів розумного дому належать: IoT-датчики, шлюзи, протоколи, мікропрограмне забезпечення, хмарні обчислення, бази даних, проміжне програмне забезпечення та шлюзи. Хмару IoT можна розділити на платформу як послуга (PaaS) та інфраструктуру як послугу (IaaS). Додаток розумного дому оновлює домашню базу даних у хмарі, щоб віддалені люди мали доступ до неї та отримували найновіший статус будинку. Типова платформа IoT містить: захист пристрою та автентифікацію пристроїв, посередників повідомлень та черги повідомлень, адміністрування пристроїв, протоколи, збір даних, візуалізацію, можливості аналізу, інтеграцію з іншими веб-службами, масштабованість, API для потоку інформації в реальному часі та бібліотеки з відкритим кодом. Датчики IoT для домашньої автоматизації

відомі своїми можливостями зондування, такими як: температура, люкс, рівень води, склад повітря, відеокамери для спостереження, голос / звук, тиск, вологість, акселерометри, інфрачервоні, вібрації та ультразвукові. Деякі з найбільш часто використовуваних датчиків розумного будинку - це датчики температури, більшість - цифрові датчики, але деякі є аналоговими і можуть бути надзвичайно точними. Датчики Люкс вимірюють світність. Ультразвукові датчики рівня води.

Датчики рівня поплавкового рівня пропонують розробникам IoT більш точну можливість вимірювання. Датчики складу повітря розробники використовують для вимірювання конкретних компонентів у повітрі: моніторинг CO, вимірювання рівня водню в газі, міра оксиду азоту, рівень небезпечних газів. Більшість із них мають час нагрівання, а це означає, що для отримання точних значень потрібен певний час. Він покладається на виявлення газових компонентів на поверхні лише після того, як поверхня буде нагріта достатньо, значення починають проявлятися. Відеокамери для спостереження та аналітики. Діапазон камер із високошвидкісним з'єднанням. Використання процесора Raspberry Pi рекомендується, оскільки його модуль камери дуже ефективний завдяки своєму гнучкому роз'єму, підключеному безпосередньо до плати.

Звукові детектори широко використовуються для моніторингових цілей, виявляючи звуки та діючи відповідно. Деякі навіть можуть виявляти наднизький рівень шуму і тонко налаштовуватися серед різних рівнів шуму.

Датчики вологості відчувають рівень вологості повітря у розумних будинках. Його точність та точність залежать від конструкції та розміщення датчика. Деякі датчики, як DHT22, побудований для швидкого прототипування, завжди будуть працювати погано в порівнянні з високоякісними датчиками, такими як HIH6100. Для відкритих просторів розподіл навколо датчика очікується рівномірним, що вимагає менших коригувальних дій для правильної калібрування.

Протоколи інтелектуального домашнього спілкування: Bluetooth, Wi-Fi або GSM. Інтелектуальні та бездротові бездротові протоколи Bluetooth з мережевими можливостями та алгоритмами шифрування даних. Zigbee - це мережевий протокол на основі частоти радіочастот для IoT. Протокол X10, який використовує електропроводку для сигналізації та управління. Insteon, бездротовий та дротовий зв'язок. Z-хвиля спеціалізується на захищеній домашній автоматизації. UPB, використовує існуючі лінії електропередач. Нитка - безоплатний протокол для автоматизованого інтелектуального дому. ANT - протокол надпотужної потужності для створення малопотужних датчиків з можливістю розподілу сітки. Переважними протоколами є Bluetooth з низькою енергією, Z-хвиля, Zigbee та нитка. Розміщення щодо включення шлюзу може включати: хмарне підключення, підтримувані протоколи, складність налаштування та підтримку прототипування. Домашнє управління складається з наступного: стан машини, шина подій, журнал обслуговування та таймер.

Модульність: дозволяє концепція пакету, динаміка виконання, програмними компонентами можна керувати під час виконання, орієнтацією на обслуговування, керувати залежностями між пакетами, рівнем життєвого циклу: контролює життєвий цикл пакетів, службові шари: визначає динамічну модель зв'язку між різними модулями , фактичні послуги: це рівень програми. Рівень безпеки: необов'язково, використовує архітектуру безпеки Java 2 та керує дозволами різних модулів.

OpenHAB - це основа, що поєднує автоматизацію дому та шлюз IoT для розумних будинків. Його особливості: правила двигуна, механізм ведення журналу та абстракція користувальницького інтерфейсу. Правила автоматизації, орієнтовані на час, настрій чи атмосферу, просту конфігурацію, звичайне підтримуване обладнання. [17]

1.4 Розробка власної концепції

Власна розробка системи розумного дому представляє собою систему моніторингу показників температури та вологості з можливістю масштабування кількості пристроїв вимірювання. Систему умовно можна розділити на дві частини: серверну, яка безпосередньо взаємодіє з користувачем та зберігає й відображає дані отримані від пристроїв; клієнтську (апаратну), що складається з певної кількості пристроїв збору показників температури та вологості. Обмін даними між серверною та клієнтською частинами відбувається з-за допомогою протоколу HTTP. Як було сказано вище, в даній бакалаврській роботі проводиться розробка саме апаратної частини системи для вимірювання показників, а також її програмне забезпечення та супутнє програмне забезпечення для полегшення встановлення необхідних налаштувань роботи пристроїв вимірювання. Вона представляє з себе платформу, що складається з мікроконтролера, датчика температури та вологості, а також (в залежності від варіанту реалізації) компоненту індикації роботи пристрою у вигляді світлодіоду або дисплею. Аналіз, порівняння та вибір цих компонентів проводиться в наступних розділах нижче.

ВИСНОВКИ ДО РОЗДІЛУ 1

Концепція інтернету речей в сучасному світі, стає все більш звичною технологією для нас.

В даному розділі було оглянуто весь розвиток «розумного будинку» від її започаткування та до формування технології вже в знайомому для нас вигляді. Проаналізувавши сучасні системи розумного дому, їх недоліки та переваги, було вирішено створити власний варіант рішення у вигляді системи моніторингу температури та вологості для розумного дому. Визначено певні критерії яким має відповідати створений пристрій.

РОЗДІЛ 2
РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ

Вибір апаратних компонентів для розробки прототипу пристрою для системи моніторингу проводиться з точки зору максимальної економічної вигоди виготовлення однієї одиниці, з забезпеченням необхідних обчислювальних можливостей, засобів зв'язку та підключення (Wi-Fi контролер, периферійні піни), легкості підключення плати до комп'ютера для налаштування та прошивки, а також естетичного вигляду в цілому.

2.1 Огляд доступних мікроконтроллерів

Мікроконтролер (або мікрокомп'ютер) є центральним елементом у розроблювальному пристрої. Саме він відповідає за керування підключеними датчиками, засобами виводу інформації та індикаторами роботи. В останнє десятиліття на ринку техніки з'явилося величезне різноманіття мікроконтролерних та мікрокомп'ютерних систем, що є універсальні за призначенням та мають невелику ціну. Враховуючи раніше вказані вимоги, доцільним буде розглянути наступні серії та моделі.

2.1.1 Серія Raspberry Pi

Серія мікрокомп'ютерів Raspberry Pi розроблюється британським фондом Raspberry Pi Foundation як бюджетні системи з ціллю стимулювання вивчення інформатики в навчальних закладах, але згодом отримали велику популярність та визнання у широкого кола аматорів для використання у власних проектах, зокрема у домашній автоматизації та медіацентрах. Даний одноплатний комп'ютер має достатньо велику кількість периферійних портів

різного призначення в залежності від моделі. Однією з найцікавіших особливостей Raspberry Pi є наявність портів GPIO, завдяки яким комп'ютер можна розширяти різноманітною кількістю модулей від сторонніх виробників або власного виробництва. На платі відсутня внутрішня енергонезалежна пам'ять, що компенсується наявністю слоту для карт формату Micro SD.

Оскільки Raspberry Pi плати являють собою повноцінні комп'ютери з процесорами на основі ARM архітектури, існує велика різноманітність операційних систем створених спеціально для них. До офіційно підтримуваних ОС відносяться:

- Raspbian – основана на Debian операційна система від Raspberry Pi Foundation
- Pidora – варіант Fedora для Raspberry Pi
- RISC OS – "рідна" операційна система для RISC-процесорів (в тому числі і для процесорів ARM)
- Windows 10 IoT – урізана версія Windows 10 спеціально для Raspberry Pi

Також існують неофіційні версії ОС Android адаптовані для мікрокомп'ютера. Офіційними мовами програмування є Python, як основна, та Scratch, для навчальних цілей та ознайомлення з мікрокомп'ютерами. У таблиці 2.1 представлено повний список усіх випущених модифікацій мікрокомп'ютерів Raspberry Pi та їх характеристики

Таблиця 2.1. Модельний ряд комп'ютерів Raspberry Pi

Дата виходу	Версія	Мікроархітектура	Частота	Кількість ядер	ОЗП	GPIO	USB	Ethernet	Wi-Fi	Bluetooth
квітень 2012	**B**	ARM1176JZ-F	700 МГц	1	512 МБ	26 пінів	2 порта	есть	—	—
лютий 2013	**A**	ARM1176JZ-F	700 МГц	1	256 МБ	26 пінів	1 порт	—	—	—
червень 2014	**B+**	ARM1176JZ-F	700 МГц	1	512 МБ	40 пінів	4 порта	есть	—	—
Листопад 2014	**A+**	ARM1176JZ-F	700 МГц	1	256 МБ	40 пінів	1 порт	—	—	—
Лютий 2015	**2B**	ARM Cortex-A7	900 МГц	4	1 ГБ	40 пінів	4 порта	есть	—	—
Листопад 2015	**Zero**	ARM1176JZ-F	1 ГГц	1	512 МБ	40 пінів	1 порт[120]	—	—	—
Лютий 2016	**3B**	Cortex-A53 (ARM v8)	1,2 ГГц	4	1 ГБ	40 пінів	4 порта	есть	802.11n	4.1
Лютий 2017	**Zero W**	ARM1176JZ-F	1 ГГц	1	512 МБ	40 пінів	1 порт	—	802.11n	4.0
березень 2018	**3B+**	Cortex-A53 (ARM v8)	1,4 ГГц	4	1 ГБ	40 пінів	4 порта	Gigabit через USB2	802.11ac	4.2
жовтень 2018	**3A+**	Cortex-A53 (ARM v8)	1,4 ГГц	4	512 МБ	40 пінів	1 порт	—	802.11ac	4.2

Серед наведеного списку різновидів Raspberry Pi для реалізації девайсу найкраще підходять моделі 3B через наявність Wi-Fi модулю та Ethernet порту та модель Zero W завдяки компактним розмірам та меншій ціні в порівнянні зі новішими моделями, при цьому вони все ще достатньо розповсюдженні на ринку, тому знайти їх у продажу не є великою проблемою. Зображення зовнішнього вигляду Raspberry Pi 3B наведені на рис. 2.1 та рис. 2.2, Raspberry Pi Zero W на рис. 2.3 та 2.4. Розміщення компонентів, периферійних портів та інтерфейсів для даних мікрокомп'ютерів відповідно зображені на рис. 2.5 та рис. 2.6.

Рис. 2.1 — Raspberry Pi Model 3B вигляд зверху [1]

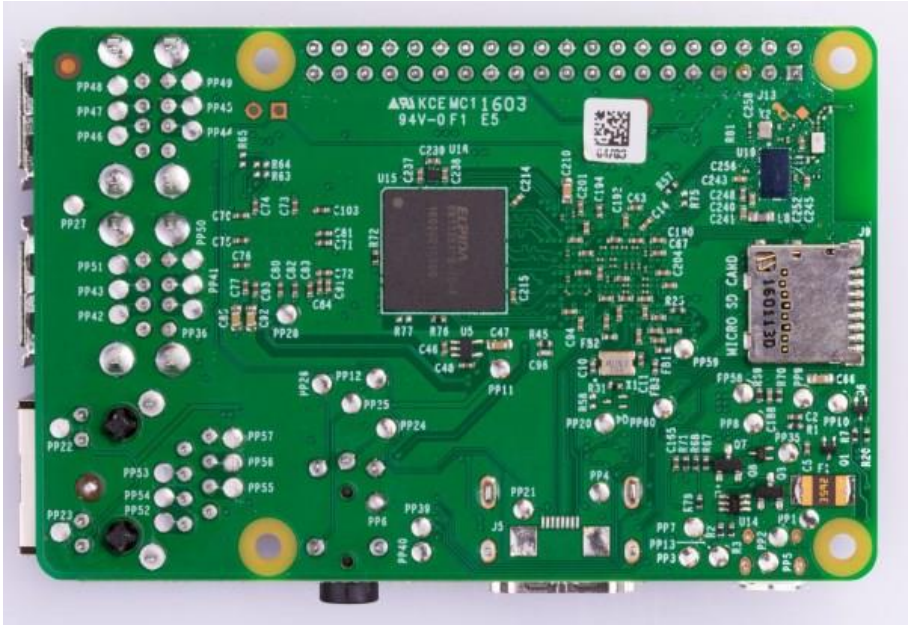


Рис. 2.2 — Raspberry Pi Model 3B вигляд знизу [1]

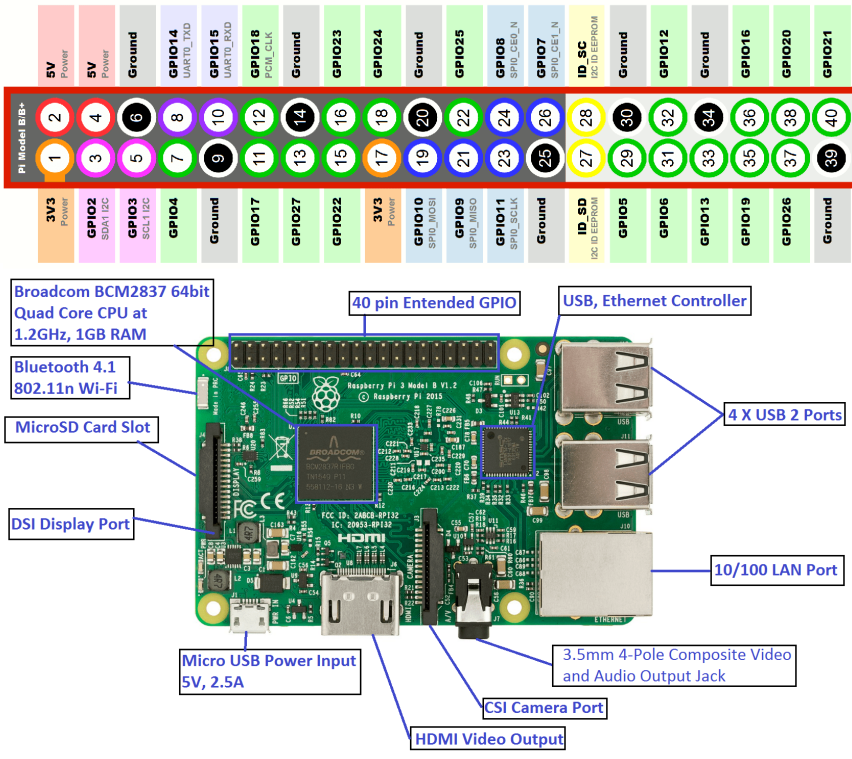


Рис. 2.3 – Розташування компонентів на платі мікрокомп’ютера моделі 3B та призначення портів GPIO [2]

Рис. 2.4 – Raspberry Pi Zero W вигляд зверху [3]

Рис. 2.5 – Raspberry Pi Zero W вигляд знизу [3]

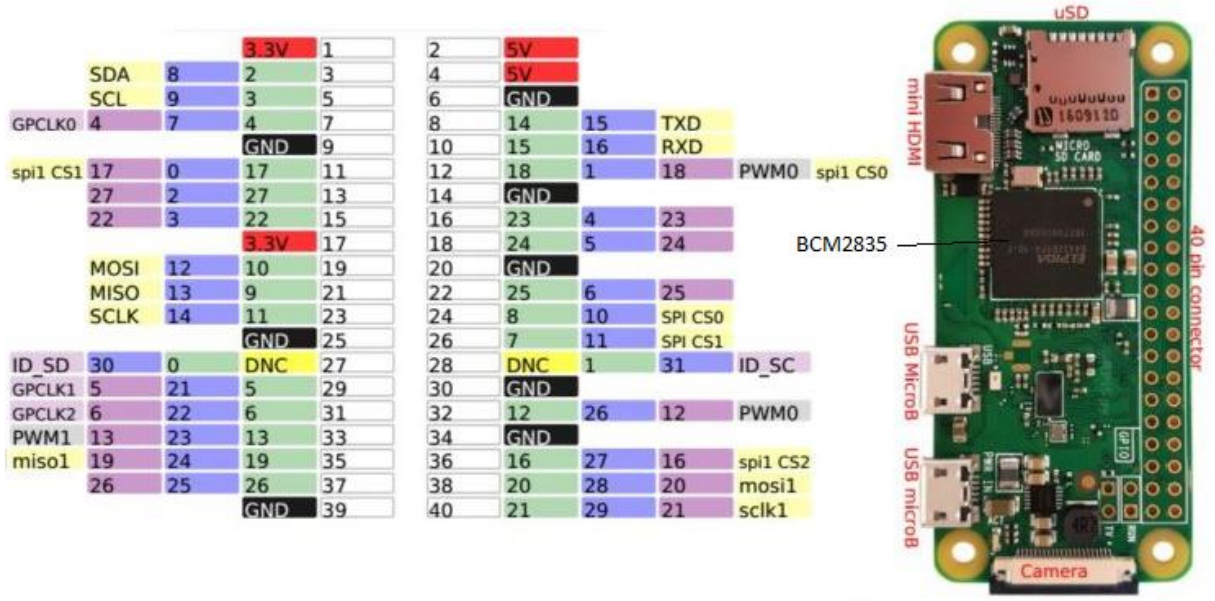


Рис. 2.6 – Розташування компонентів на платі мікрокомп’ютера моделі Zero W та призначення портів GPIO [4]

2.1.2 Серія Arduino

Серія апартно-програмних засобів для створення простих систем автоматизації та робототехніки, більше орієнтована для прототипування та використання в непрофесіональних проектах. Переважна більшість плат Arduino побудована на 8-бітних мікроконтролерах AVR архітектури від компанії Atmel (ATmega8, ATmega168, ATmega328, ATmega1280, ATmega2560) з великою різноманітністю варіантів кількості вбудованої пам`яті, пінів для розширення та ін. Майже всі плати обладнані мінімально необхідним набором обв`язок для нормальної роботи (стабілізатор напруги, кварцовий резонатор та т. п.). Мікроконтролери Arduino мають презаписаний завантажувач (bootloader), що полегшує завантаження програм на встановлену флеш-пам'ять, без необхідності використання традиційних окремих апаратних програматорів. Завантажувач з'єднується з комп'ютером через інтерфейс USB (при його наявності) або з-за допомогою окремого перехідника UART. Оригінальні плати Arduino або сумісні з ними спроектовані з врахуванням їх легкого розширення з-за допомогою спеціальних компонентів (так званих шилдів) від різних виробників.

Програмне забезпечення для мікроконтролерів створюється з-за допомогою мов програмування C або C++. Крім того, існує спеціально створене інтегроване середовище розробки Arduino, що складається з редактору коду, набору базових компіляторів (включаючи основний AVR-GCC) та засобів завантаження прошивки на плати. При цьому для створення так званих «скетчів» використовується C-подібна мова програмування Wiring, що в подальшому компілюється AVR-GCC.

Таблиця 2.2 представляє список актуальних моделей плат Arduino з основними характеристиками. На рисунках 2.7, 2.8, 2.9 та 2.10 зображенні найпопулярніші моделі плат.

Таблиця 2.2. Моделі плат Arduino

Arduino	Процесор	Напруга живлення	Флеш-пам'ять, КБ	EEPROM, КБ	SRAM, КБ	Двійкові входи/виходи	… з ШІМ	Аналогові входи	Інші інтерфейси	Розміри, мм
Diecimila	ATmega168	5 В	16	0.5	1	14	6	6		68.6 × 53.3
Due	ATMEL SAM3U		256	0	50	54	16	16		
Duemilanove	ATmega168/328P	5 В	16/32	0.5/1	1/2	14	6	6		68.6 × 53.3
Fio	ATmega328P	3.3 В	32	1	2	14	6	8		40.6 × 27.9
Leonardo	Atmega32u4	5 В	32	1	2	14	6	12		68.6 × 53.3
Mega	ATmega1280	5 В	128	4	8	54	14	16		101.6 × 53.3
Nano	ATmega168 or ATmega328	5 В	16/32	0.5/1	1/2	14	6	8		43 × 18
Pro Mini	ATmega328P	5 В або 3.3 В	32	1	2	14	6	8	UART	33 × 18
Uno	ATmega328P	5 В	32	1	2	14	6	6		68.6 × 53.3

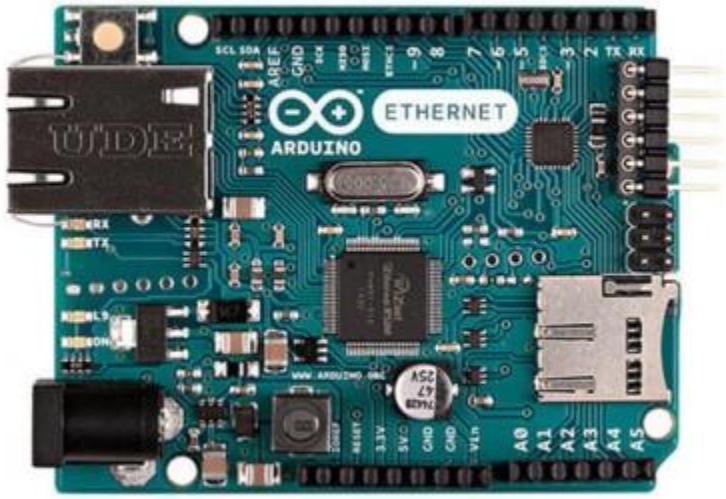


Рис. 2.7. Плата Arduino Ethernet R3. Вигляд зверху. [5]

Рис. 2.8. Плата Arduino Uno. Вигляд зверху. [6]

Рис. 2.9. Плата Arduino MKR WiFi 1010. Вигляд зверху. [7]

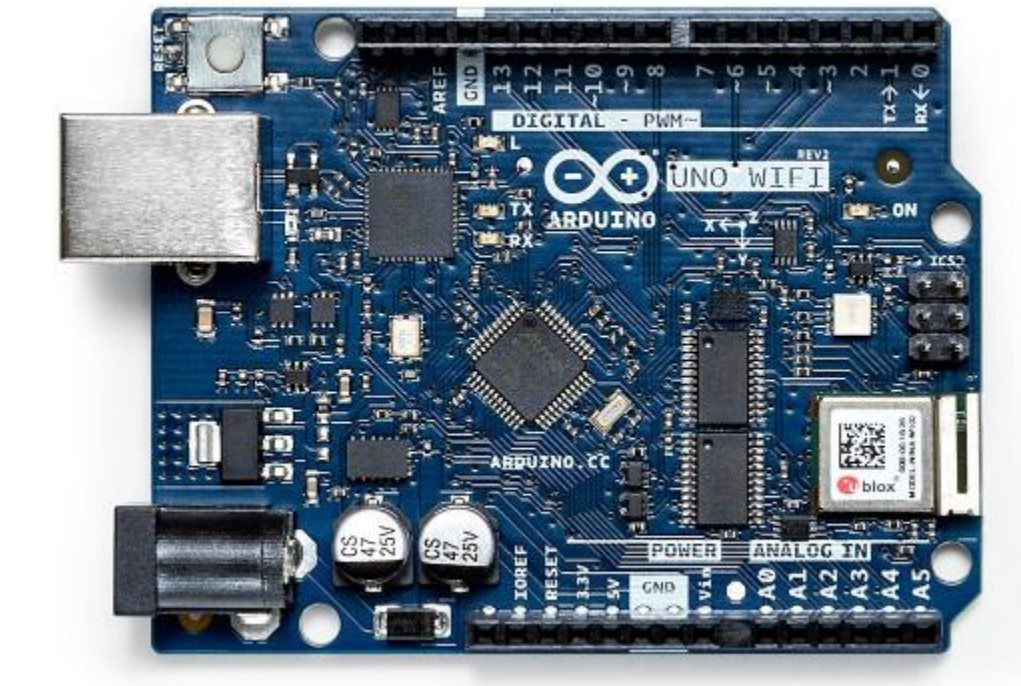


Рис. 2.10. Плата Arduino Uno WiFi Rev 2. Вигляд зверху. [7]

2.1.3 Серія ESP

ESP32 та ESP8266 представляють собою, так звані системи на кристалі (SoC) з інтегрованими Bluetooth (лише для ESP32) та Wi-Fi контролерами від китайського виробника Espressif Systems. Дані мікроконтролери, крім наявності безпровідних можливостей, відзначаються низьким енергоспоживанням та невеликою ціною, саме тому, вони системи отримали величезну популярність у **любительських** проектах розумного дому та системах автоматизації.

ESP8266

Мікроконтролер вперше з'явився у 2014 році і зразу привернув до себе увагу завдяки незвично низькій ціні. Оскільки мікроконтролер не має на кристалі вбудованої енергонезалежної пам'яті, виконання програми проводиться зі зовнішнього ПЗУ під'єднаного по інтерфейсу SPI. Заявлена підтримка до 16 мегабайт зовнішньої пам'яті програм.

Основні характеристики та можливості ESP8266:

- 80 MHz 32-bit процесор Tensilica Xtensa L106. Існує можливість розгону до 160 MHz.
- IEEE 802.11 b/g/n Wi-Fi. Існує підтримка WEP и WPA/WPA2.
- 14 портів вводу-виводу (серед них програмісту доступні лише 11), підтримка інтерфейсів SPI, I²S, UART, I2C, наявність 10-bit АЦП
- Живлення від 2,2 до 3,6 В. Споживання струму до 215 мА в режимі передачі, до 100 мА в режимі прийому, до 70 мА в режимі очікування. Підтримуються три режими зниженого використання енергії, але без зберігання підключення до точки доступу мережі: Modem sleep (15 мА), Light sleep (0.4 мА), Deep sleep (15 мкА)

Весною 2016 року Espressif запустив у масове виробництво мікросхеми ESP8285, що вміщує на одній мікросхемі як систему ESP8266, так і 1 мегабайт флеш-пам'яті.

ESP32

Восени 2015 року Espressif представила подальший розвиток лінійного ряду ESP – систему ESP32.

Основні характеристики та можливості ESP8266:

- 160-240 MHz 32-bit процесор Tensilica Xtensa LX6 з одним або двума ядрами в залежності від версії.
- 448 Кб ПЗУ, 520 Кб ОЗУ. Зовнішні ОЗУ/ПЗУ підключені по SPI інтерфейсу, заявлена можливість підключення до 64 Мб.
- Живлення аналогічне ESP8266 (від 2.2 до 3.6 В)
- Wi-Fi 802.11, Bluetooth v4.2 (в тому числі Low Energy).
- Збільшена кількість портів та периферії: ADC, DAC, 4 SPI, 2 I2S, 3 UART, CAN. Інтерфейс для SD карт. Ethernet MAC

Даний мікроконтролер дійсно можна назвати великим кроком вперед в порівнянні з попередньою моделлю. Зміни в кращу сторону прослідковуються у всіх характеристиках. Найбільш значущі відміності відображені на рис. 2.11.

	ESP8266	ESP32
MCU	Xtensa Single-core 32-bit L106	Xtensa Dual-Core 32-bit LX6 with 600 DMIPS
802.11 b/g/n Wi-Fi	HT20	HT40
Bluetooth	No	Bluetooth 4.2 and BLE
Typical Frequency	80 MHz	160 MHz
SRAM	No	Yes
Flash	No	Yes
GPIO	17	36
Hardware /Software PWM	None / 8 channels	None / 16 channels
SPI/I2C/I2S/UART	2/1/2/2	4/2/2/2
ADC	10-bit	12-bit
CAN	No	Yes
Ethernet MAC Interface	No	Yes
Touch Sensor	No	Yes
Temperature Sensor	No	Yes
Hall effect sensor	No	Yes
Working Temperature	-40°C to 125°C	-40°C to 125°C

Рис. 2.11. Порівняльна таблиця ключових характеристик ESP8266 та ESP32.

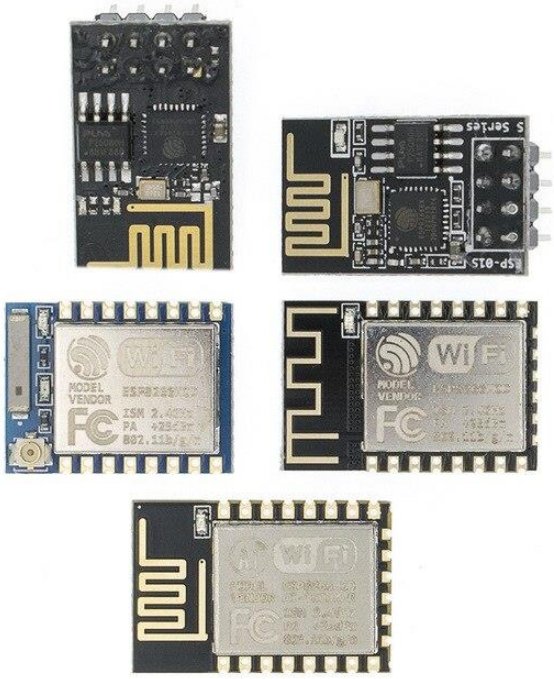


Рис. 2.12. Мікроконтролери ESP8266 різного покоління та версій. [9]

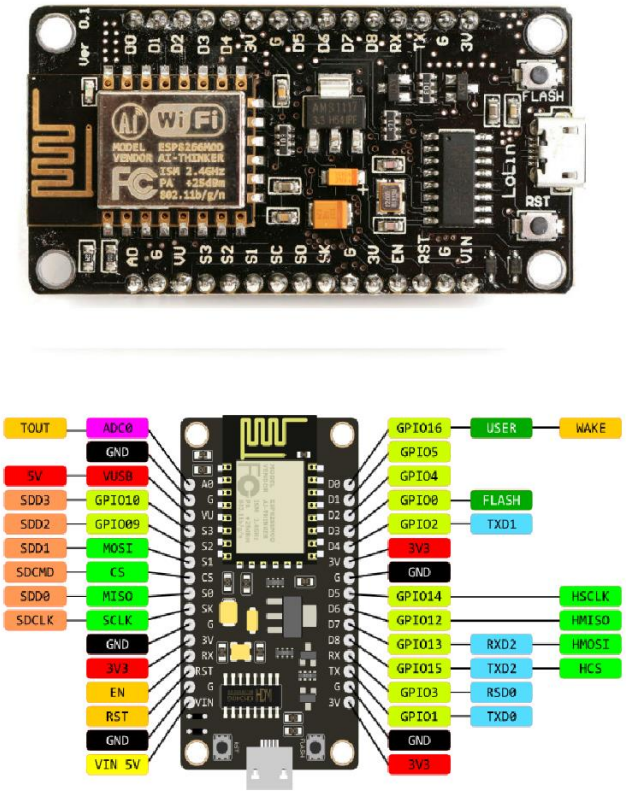


Рис. 2.13 Плата NodeMCU v3 на основі ESP8266 та її розпінування. [10]

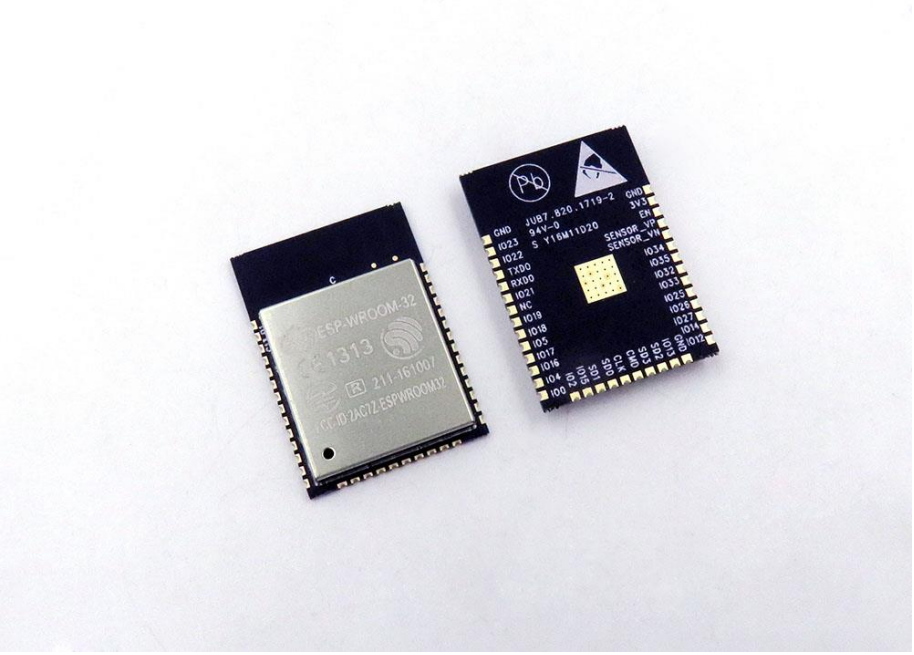


Рис. 2.14 Мікроконтролер ESP32. [9]

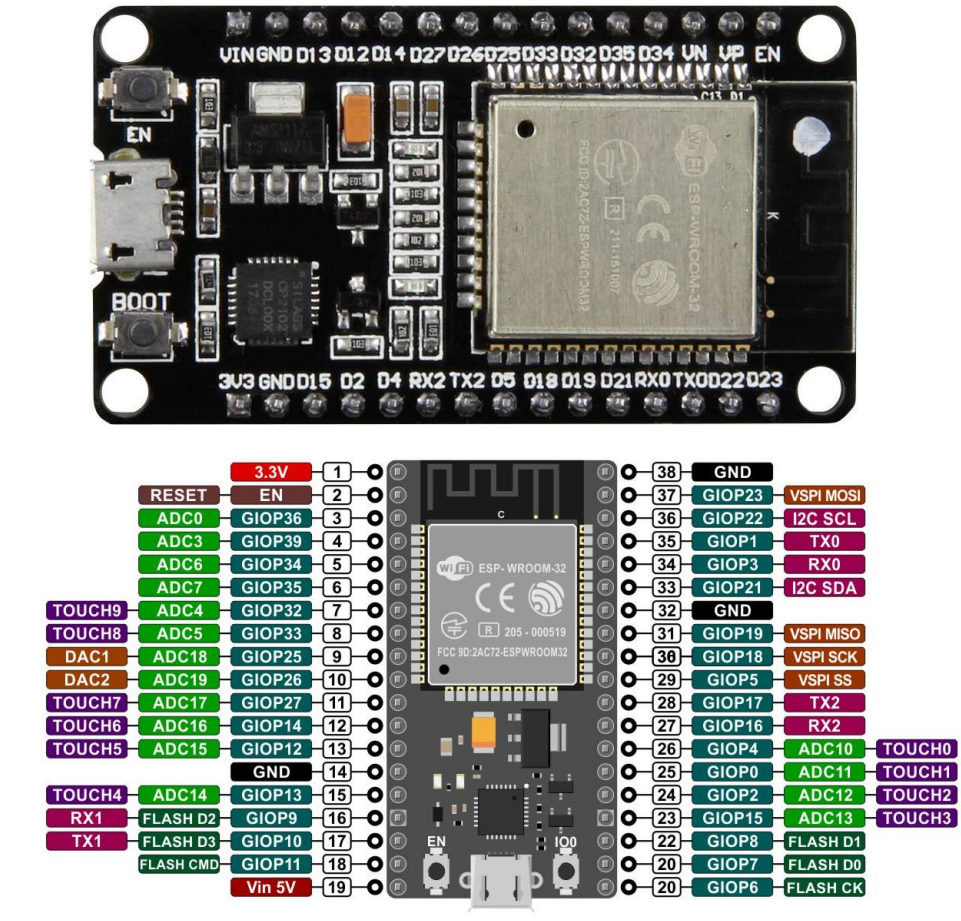


Рис. 2.15. плата NodeMCU-ESP32 та її розпінування. [11]

2.1.4 Висновок щодо мікроконтролера

Розглянуто деякі моделі мікроконтролерів, що найкраще **підходять** для реалізації цільового продукту.

Мікрокомп’ютери Raspberry Pi 3B та Raspberry Pi Zero W мають найкращі показники продуктивності та повний набір периферійних інтерфейсів, але в той же час, обчислювальна потужність є надлишковою при використанні в даному проекті та не зможе бути реалізована в повному обсязі. Найбільшим недоліком в даній ситуації мікрокомп’ютерів Raspberry Pi є ціна даних моделей, яка перевищує конкурентів в декілька разів.

Серія Arduino це можливо найпопулярніший варіант вибору для власноручно створених проектів в світі. Вони мають невелику ціну, обчислювальної потужності достатньо для використання в подібних системах, а завдяки великій розповсюдженості проблем з програмуванням та пошуком бібліотек не має виникнути. Але мікроконтролери Arduino не мають вбудованих засобів зв’язку, як наприклад WiFi чи Ethernet адаптери, що змушує до залучення додаткових сторонніх модулей, що в свою чергу збільшує складність одного пристрою та його ціну.

Системи ESP8266 та ESP32 є найкращим вибором для створення системи моніторингу для розумного дому. Це достатньо молоді мікроконтролери, що поєднують в собі чудову продуктивність, наявність периферійних інтерфейсів та модулів безпровідного зв’язку та найголовніше вони мають невелику ціну. Для реалізації прототипу була обрана система на кристалі ESP8266, що в порівнянні ESP32 відрізняється відсутністю Bluetooth, який не використовується в даному випадку, та меншою обчислювальною потужністю, тим не менш якої достатньо, при цьому має перевагу в ціні.

2.2 Вибір датчика

Системи моніторингу для розумного дому різняться за параметрами моніторингу в залежності від підключених сенсорів. В даній роботі розглядається робота з датчиками температури та вологості, через їх велику розповсюдженість, невелику вартість та легкість підключення. На ринку представлено велику різноманітність модулів для вимірювання температури та тиску. Розглянуто сенсори сімейства DHT, а саме DHT11, DHT21 та DHT22. Підключення усіх сенсорів серії DHT відбувається по інтерфейсу 1-Wire. В таблиці 2.3 представлено порівняння основних характеристик розглянутих датчиків. Для розробки пристрою було обрано датчик DHT21 через можливість точнішого виміру температури та вологості в порівнянні з DHT11 та наявності кращого корпусу для захисту мікросхеми сенсору в порівнянні з DHT22. Зображення розглянутих датчиків представленні на рис. 2.13 та 2.14.

Таблиця 2.3. Таблиця порівняння датчиків вимірювання температури та вологості.

	DHT11	DHT21	DHT22
Допустимий діапазон вимірювання температури	0 … 50 °C	-40 … +80 °C	-40 … +80 °C
Похибка вимірювання температури	± 2%	± 0.5%	± 0.5%
Допустимий діапазон вимірювання вологості	20 .. 95 %	0 .. 99.9%	0 .. 99.9%
Похибка вимірювання вологості	4-5 %	2-5 %	2-4 %
Вага	6 г	16 г	9 г
Габарити	14 × 30 × 10 мм	59 × 27 × 14 мм	15 × 12 × 6 мм
Мінімальний час між вимірюванням показників	1 сек.	2 сек.	2 сек.

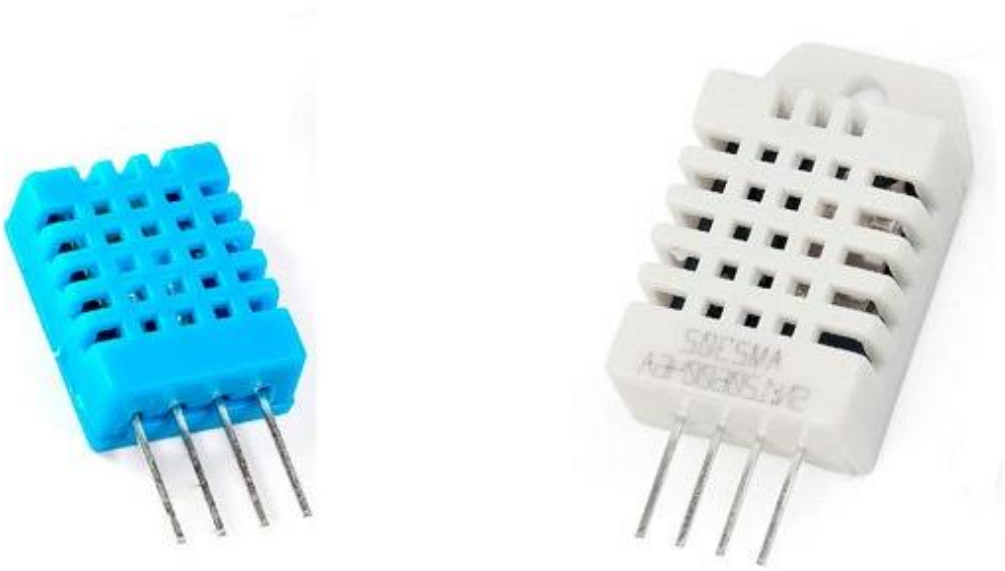


Рис. 2.12. датчики DHT11 (справа) та DHT22 (зліва). [12]

Рис. 2.12. датчик DHT21. [12]

2.3 Засоби індикації роботи та відображення показань

Для відображення стану роботи пристрою передбачено підключення світлодіоду, який з-за допомогою певного світлового сигналу відображає роботу. Перелік наявних світлових сигналів буде визначений вже при створенні прошивки. В рамках даної роботи був обраний звичайний світлодіод зеленого кольору зображений на рис. 2.14.

Також існує можливість підключення OLED дисплею для відображення інформації безпосередньо на пристрої. Ця функція не ж обов’язковою та на загальну працездатність пристрою не впливає, лише при наявності додаткових коштів підвищує зручність користування. Варто зауважити, що для готового виробу був використаний лише світлодіод без дисплею.

Використаний у роботі дисплей має діагональ 0.96 дюйма, роздільність складає 128 на 64 пікселів. Підключення дисплею відбувається по інтерфейсу I2C. Зовнішній вигляд зображений на рис. 2.15.

Рис. 2.12. світлодіод.

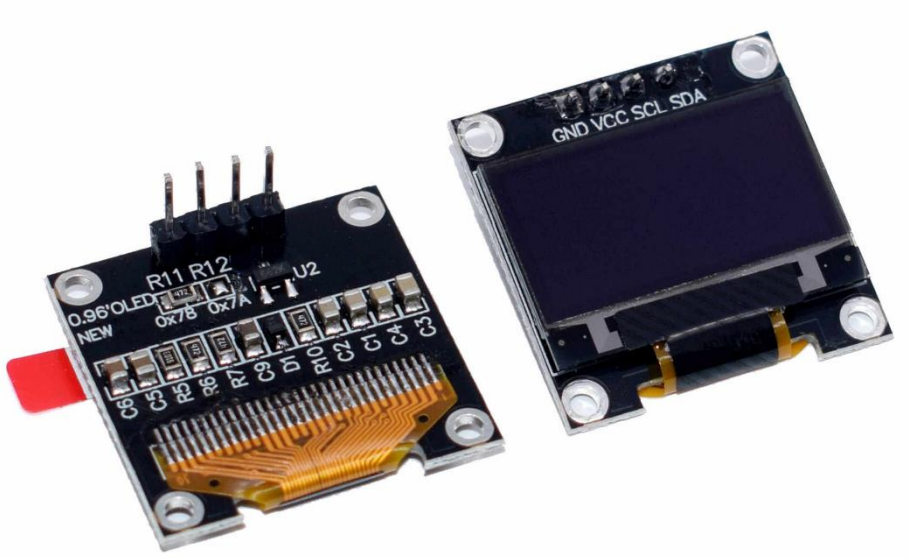


Рис. 2.12. OLED дисплей [13]

2.4 Корпус виробу

Корпус використаний у роботі був придбаний у приватного виробника. Виготовлений з використанням 3D-принтеру, створений спеціально для захисту плати NodeMCU ESP8266 та складається з двох частин. З бокових сторін наявні спеціальні отвори, для закріплення виробу на гвинтах. На передній частині отвір для прокладення проводів підключення датчика температури та вологості з платою. На задній частині передбачений отвір на рівні micro-USB порту NodeMCU для прошивки та живлення виробу. На верхній частині корпусу наявний отвір для світлодіоду. Зображення корпусу представлені на рис. 2.13, 2.14 та 2.15.

Рис. 2.13. Розібраний корпус.

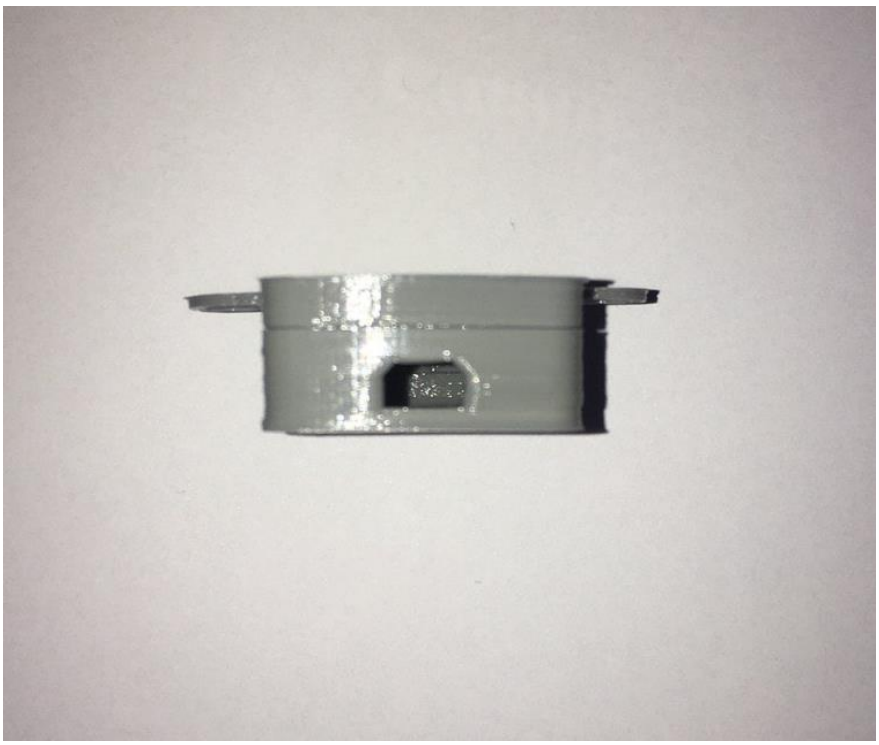


Рис. 2.14. Передня частина корпусу.

Рис. 2.15. Верхня частина корпусу.

2.5 Схема тестового стенду та реалізація прототипу

Для реалізації тестового стенду була використана макетна плата та набір з'єднувальних проводів типу тато-тато. Для створення стенду були використані наступні компоненти:

- NodeMCU v3 на базі ESP8266
- Датчик температури та вологості DHT21
- Дисплей SSD1306
- світлодіод

Компоненти підключаються до NodeMCU наступним чином:

- Датчик температури:
  - Червоний провід живлення до +5 В
  - Чорний провід землі до GND
  - Жовтий провід до піну D1
- Дисплей:
  - Пін GND до піну GND контролера
  - Пін VDD до +5 В
  - Пін SCK до піну D6
  - Пін SDA до піну D5
- Світлодіод:
  - Нога землі до GND
  - Нога живлення до +5 В

Живлення всієї системи відбувається через micro-USB порт мікроконтролера. Після створення прошивки та налаштування необхідних параметрів, тестування працездатності, всі частини були спаяні та вкладені у корпус.

Зібраний тестовий стенд зображений на рис. 2.16. Схема тестового стенду зображена на рис. 2.17. Зібраний пристрій зображений на рис. 2.18.

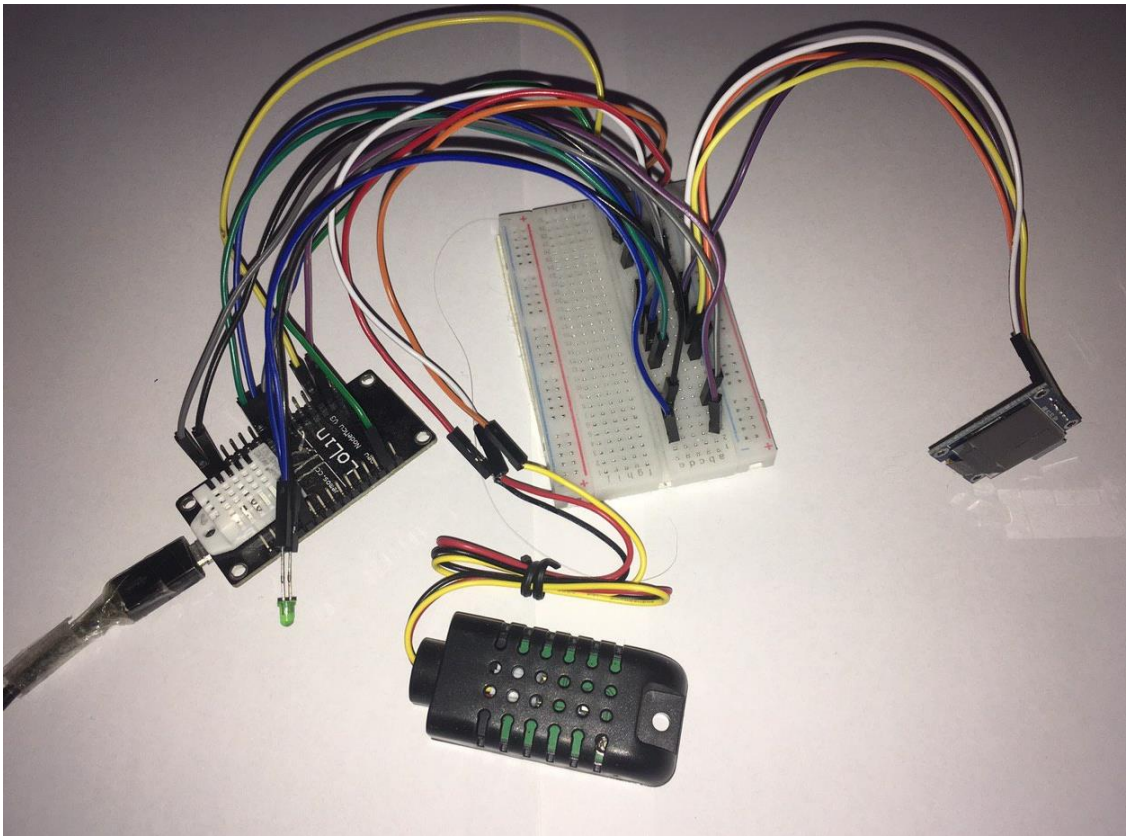


Рис. 2.15. Тестовий стенд.

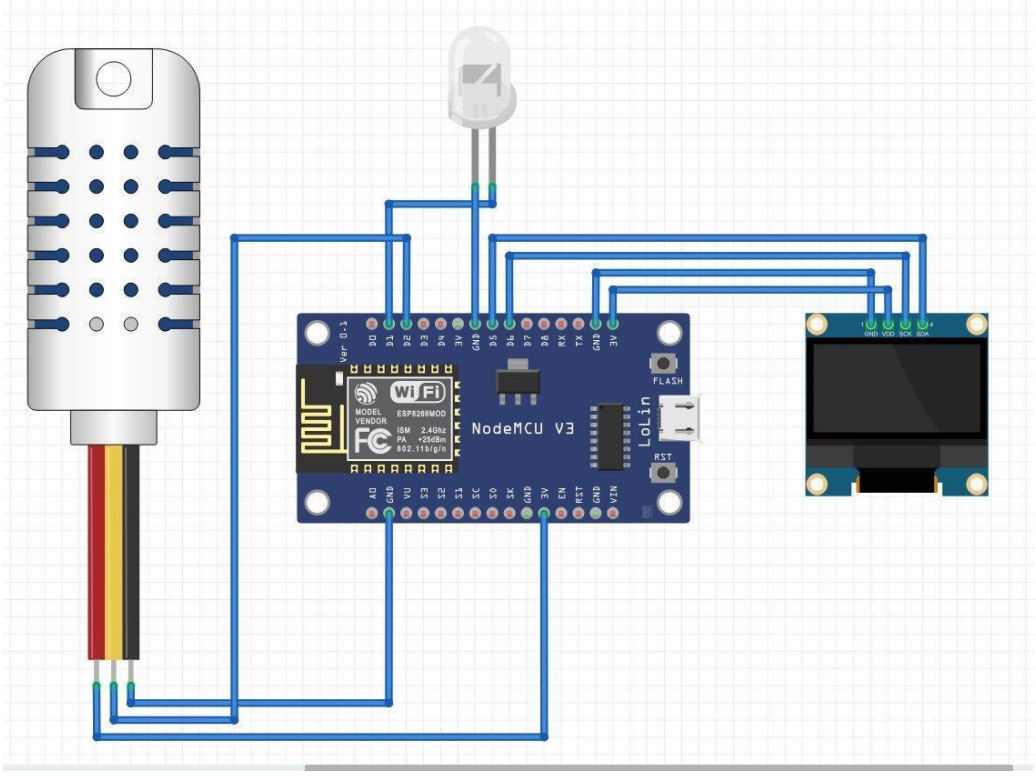


Рис. 2.16. Схема тестового стенду.

Рис. 2.15. Зібраний робочий прототип.

ВИСНОВОК ДО РОЗДІЛУ 2

Для реалізації прототипу пристрою системи моніторингу для розумного дому було розглянуто перелік необхідних компонентів та обрано найоптимальніші варіанти.

В якості мікроконтролеру було обрано ESP8266 на платформі NodeMCU v3, який виділяється серед конкурентів невеликою ціною та наявністю Wi-Fi модулю.

Для виміру показів температури та вологості було обрано датчик DHT21.

Для відображення інформації було обрано OLED дисплей та для відображення роботи пристрою обраний світлодіод.

З-за допомогою макетної плати розроблено початковий робочий прототип. Після налаштування та перевірки працездатності компоненти були спаяні та поміщені у корпус.

РОЗДІЛ 3

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для роботи створеного прототипу необхідно створити відповідне програмне забезпечення. Останнє має виконувати наступні функції:

- підключення пристрою до мережі;
- збір даних з датчика температури та вологості:
- блимання світлодіоду;
- відображення інформації на дисплеї.

Створений програмний продукт має відповідати певним вимогам якості та стабільності роботи.

3.1 Огляд доступних засобів розробки

Для обраного мікроконтролеру існує достатньо великий спектр засобів для створення програмного забезпечення.

Програмні засоби розробки (програмний комплект розробника, SDK) складаються з:

- Компілятора. Компілятор для Xtensa LX106 входить в пакет компіляторів GNU Compiler Collection. Компілятор має відкриті вихідний код. У різних SDK можуть міститися різні збірки цього компілятора, трохи відрізняються підтримуваними опціями.
- Бібліотек для роботи з периферією контролера, стеків протоколів WiFi, TCP / IP.
- Коштів завантаження файлу в пам'ять програм мікроконтролера.

Espressif вільно поширює свій комплект розробника. У цей комплект входить компілятор GCC, бібліотеки Espressif і завантажувальна утиліта XTCOM.

Бібліотеки поставляються в вигляді скомпільованих бібліотек, без вихідних текстів. Espressif підтримує дві версії SDK: одна на основі RTOS, інша на основі зворотних викликів (callback).

Крім офіційної SDK існує ряд проектів альтернативних SDK. Ці SDK використовують бібліотеки Espressif або пропонують власний еквівалент бібліотек Espressif, отриманий методами реверсінжінірінга.

- «Esp-open-sdk». Покращена версія SDK від Expressif. Містить GCC компілятор і деякі бібліотеки Expressif. Тільки Лінукс. По-русски трохи тут.
- «Unofficial Development Kit» Михайла Григор'єва. У комплект входить Windows-інсталятор, компілятор GCC власної збірки з інтеграцією з графічної IDE Eclipse, актуальні комплекти бібліотек і документації Espressif, деякі утиліти. Є російськомовний форум.
- «Arduino IDE for ESP8266» - додаток до IDE Arduino, що дозволяє програмувати ESP8266 так само легко як будь-які інші модулі Ардуіно. При цьому доступна мережева функціональність ESP8266. Компілятор GCC, завантажувач прошивки ESPTool. Детальний російськомовне опис процесу установки і доступного API тут, приклад роботи тут.
- «GNU toolchain for esp8266». Має можливість інтеграції в Visual Studio.
- «ESP8266 GCC Toolchain» Макса Філіппова.
- «Sming» [9] - проект додавання Arduino сумісних бібліотек поверх стандартних бібліотек Espressif, але без препроцесора Ардуіно (тобто програмування йде на чистому Сі).

Прошивки

Щоб спростити використання мікроконтролера в типових проектах можливе використання готових бінарних файлів, придатних до прямої заливці в ПЗУ модулів (так званих прошивок).

Готові прошивки можна розділити на кілька груп відповідно до концепції їх використання:

- Прошивки для роботи під керуванням зовнішнього контролера. У цих прошивках реалізовано завдання параметрів роботи через зовнішній контролер UART. До таких прошивка відноситься:
    - Прошивка з керуванням AT-командами з SDK Espressif. Там же можна знайти документи «4A-ESP8266__AT Instruction Set» і «4B-ESP8266__AT Command Examples». Російськомовний довідник по AT командам тут. Також існує утиліта, що дозволяє настроїти модуль без знання AT команд.
- Прошивки з вбудованими інтерпретаторами різноманітних мов високого рівня. Ці прошивки дозволяють довантажувати через UART і виконувати скрипти розробника пристрою.
    - NodeMCU (англ.) Рос. - проект на основі скриптового мови Lua. Прошивка вміє виконувати Lua-скрипти як з UART (аналогічно AT-командам) так і з внутрішньої flash пам'яті. Для завантаження скриптів в flash пам'ять підтримується файлова система. З боку Wi-Fi є вбудовані MQTT протокол і HTTP сервер. Вбудована графічна оболонка, що дозволяє підключати до ESP8266 графічні індикатори. Короткий російськомовний огляд можна знайти тут і тут. Російськомовне опис API мови тут. Офіційна документація тут. Є живі російськомовний і англомовний форуми. В інтернет-магазинах популярні недорогі модулі з прошивкою NodeMCU.
    - Espruino - проект на основі скриптового мови JavaScript. Є графічна оболонка (IDE) для роботи з вихідними текстами скриптів і закачування їх в модулі.

- ESP8266 BASIC - проект на основі скриптового мови Basic. Проект відрізняє можливість редагування і запуску скриптів через браузер, через HTML сторінку ESP8266. Також проект підтримує свою графічну утиліту для прошивки флеш-пам'яті. Підтримуються модулі з 512 кБ, 1, 2 або 4 МБ пам'яті.
- Smart.js - проект на основі скриптового мови JavaScript. Російськомовний огляд тут.
- NodeLUA - проект на основі скриптового мови Lua.
- ZBasic - проект на основі скриптового мови Basic.
- esp-lisp - уповільнений проект на основі скриптового мови Лісп.
- MicroPython - проект на основі скриптового мови MicroPython.

3.2 Робота з MicroPython

MicroPython – реалізація мови програмування Python для мікроконтролерів. Це чудова можливість для знавців цієї мови, використовуючи зручний синтаксис мови, взаємодіяти з невеликими обчислювальними пристроями. Найбільше поширення MicroPython зазнав в якості інструменту для швидкого прототипування та налаштування невеликих мікроконтролерних систем (особливо з використанням REPL). Хоч і швидкість реакції на зміни у системі достатньо велику при використанні даного дистрибутиву Python, цей параметр в більшості випадків не є критичним, що дозволяє розглядати мову, як повноцінний засіб для створення серйозних рішень у сфері вбудованих систем.

Для завантаження інтерпретатору MicroPython на плату NodeMCU використовується консольна утиліта esptool створена Espressif. Інсталяція відбувається з-за допомогою пакетного менеджеру pip, наступною командою:

```
$: pip install esptool
```

Інтерпретатор MicroPython представлений у вигляді бінарного файлу, який можливо завантажити з офіційного сайту micropython.org. На момент написання роботи, актуальною є 12 версія прошивки. Прошивка плати відбувається наступною командою, де DEVICE_PATH шлях до файлу пристрою, а MICROPYTHON_BIN шлях до прошивки:

$: esptool.py --port DEVICE_PATH --baud 460800 write_flash --flash_size=detect 0 MICROPYTHON_BIN

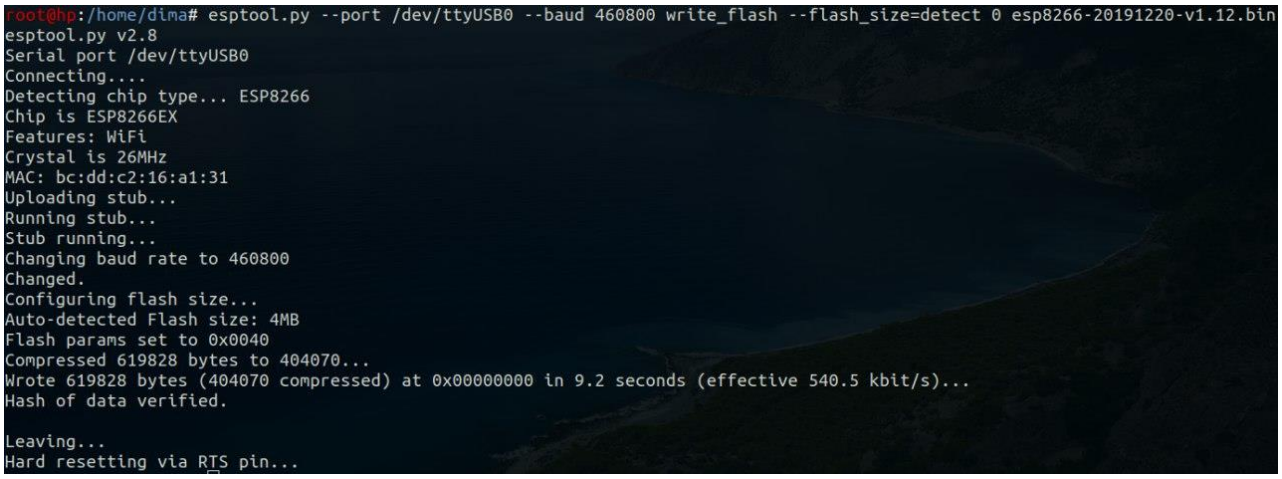


Рис. 3.1. Успішна прошивка плати NodeMCU.

При успішному завантаженні прошивки, використавши будь-який емулятор терміналу, можна підключитись по послідовному порту до плати з бодрейтом 112500. При цьому відкриється інтерактивна консоль Micropython.

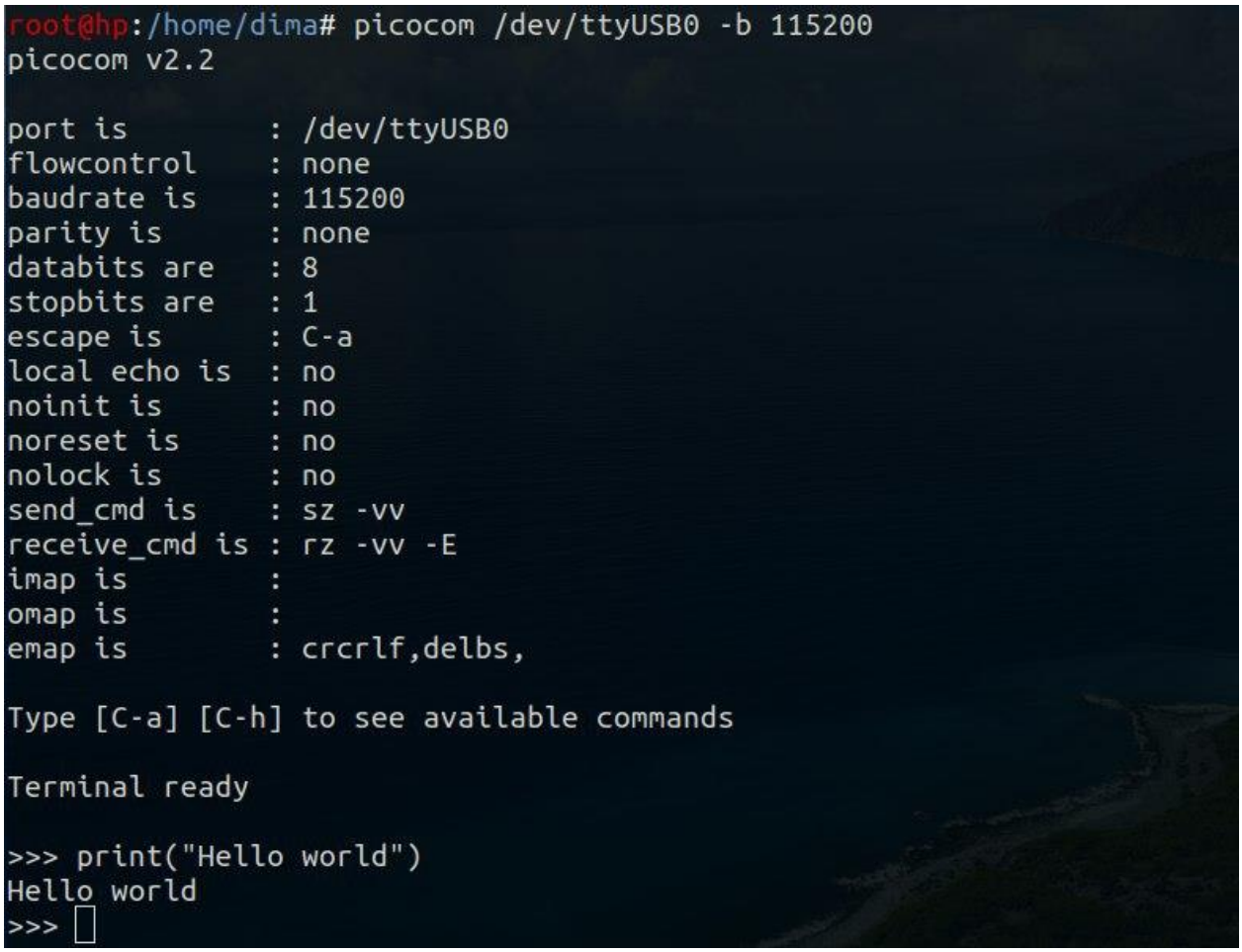


Рис. 3.2 Підключення до прошитої NodeMCU по UART.

3.3 Створення програмних файлів

При старті мікроконтролера з прошивкою MicroPython, перевіряється наявність двох файлів у корені файлової системи. Ці файли, що містять код MicroPython, виконуються кожен раз при завантажені або перезавантажені плати. Цими файлами є:

- boot.py – файл, що запускається лише один раз при завантажені прошивки та зазвичай містить низькорівневий код, що використовується для встановлення початкових параметрів системи.
- main.py – файл, що запускається після boot.py та зазвичай містить основну програму мікроконтролера.

Програма розробленого прототипу має наступну структуру:

- dht.py – файл, для взаємодії датчика температури та вологості с іншими компонентами;
- http_client.py – файл, з HTTP клієнтом для відправлення на сервер отриманих даних з сенсору;
- led.py – файл, для управління роботою світлодіоду;
- wifi.py – файл, з засобами для управління Wi-Fi модулем;
- logger.py – файл, для логування усіх виконаних дій на пристрої, для легшої відладки та діагностики проблем;
- utils.py – збірний файл, з загальними функціями для більшості компонентів;
- application.py – керуючий файл, для управління всіма модулями;

Файл керування DHT21

Клас *DHTSensor* відповідає за керування підключеним датчиком температури та вологості, а саме вимір температури та вологості, читання останніх вимірів. Нижче зображені методи, що реалізовують ці функції. З-за допомогою декораторів, методи temperature та humidity працюють як поля класу.

```
@property
def temperature(self):
    self.logger.debug('Temperature requested')
    return self.last_temperature

@property
def humidity(self):
    self.logger.debug('Humidity requested')
    return self.last_humidity

def measure(self):
    self.dht_sensor.measure()
    self.last_temperature = self.dht_sensor.temperature()
    self.last_humidity = self.dht_sensor.humidity()
    self.logger.debug('Made measure')
```

Файл керування світлодіодом

Файл містить функції для ввімкнення, вимикнення світлодіоду, блимання різної довжини, функція індикації підключення пристрою до безпровідної точки доступу.

```
def led_on():
    led.on()
    logger.debug('ON')


def led_off():
    led.off()
    logger.debug('OFF')


def led_blink(blink_time):
    logger.debug('LED blink with {} time'.format(blink_time))
    led_on()
    time.sleep(blink_time)
    led_off()


def led_multiple_blink(blink_time,
                       blink_amount):
    for _ in range(blink_amount):
        led_on()
        time.sleep(blink_time)
        led_off()


def wifi_connecting_blinker():
    logger.debug('WiFi connecting blink started')
    while True:
        led.on()
        time.sleep(0.2)
        led.off()
        is_connected = (yield)
        if is_connected:
            break
    logger.debug('WiFi connecting blink finished')
```

Файл HTTP клієнту

Клас *HTTPClient* виконує функцію відправлення показників температури та вологості на сервер. Для відправлення даних формується POST запит, в тілі якого у вигляді JSON-структури передаються показники температури та вологості.

```
class HTTPClient:

    def __init__(self, server_address, server_port):
        self.server_address = server_address
        self.server_port = server_port
        self.url = 'http://{}:{}/device{}'.format(self.server_address,
                                                  self.server_port,
                                                  DEVICE_ID)

        self.logger = Logger(name='HttpClient',
                             log_level=GLOBAL_LOG_LEVEL)

    def send_sensors_info(self, temperature, humidity):
        params = ujson.dumps({'temperature': temperature,
                              'humidity': humidity})
        response = urequests.post(self.url,
                                  headers={'Content-Type': 'application/json'},
                                  data=params)
        self.logger.info('Send sensor info to {}'.format(self.server_address))
        led_blink(0.5)
```

Файл Wi-Fi клієнту

Клас WiFiHandler відповідає за підключення пристрою до безпровідної мережі. Наявні методи активації та деактивації Wi-Fi модуля, підключення та відключення від мережі та перевірка чи є підключеним пристрій до мережі. Поточний статус Wi-Fi модуля, один з наступних:

- STAT_IDLE – модуль не підключений до бездротової мережі;
- STAT_CONNECTING – триває підключення;
- STAT_WRONG_PASSWORD – помилка підключення, невірний ключ доступу;

- STAT_NO_AP_FOUND – помилка підключення, неправильна назва точки доступу;
- STAT_CONNECT_FAIL – помилка підключення з невідомої причини;
- STAT_GOT_IP – модуль підключений

```
class WiFiHandler:

    def __init__(self, ssid, password):
        self.ssid = ssid
        self.password = password

        self.wlan = network.WLAN(network.STA_IF)
        self.wlan.active(True)

        self.logger = Logger(name='WiFiHandler',
                             log_level=GLOBAL_LOG_LEVEL)
        self.logger.debug('WiFiHandler instance is created')

    def __del__(self):
        self.wlan.active(False)
        self.logger.debug('WifiHandler instance is deleted')

    @property
    def is_connected(self):
        return self.wlan.isconnected()

    @property
    def is_active(self):
        return self.wlan.active()

    @property
    def status(self):
        return self.wlan.status()

    def deactivate(self):
        self.wlan.active(False)
        self.logger.debug('Deactivated')

    def activate(self):
        self.wlan.active(True)
        self.logger.debug('Activated')

    def disconnect(self):
        self.wlan.disconnect()
        self.logger.info('Disconnected')
```

```
    def connect(self):
        wifi_blinker = wifi_connecting_blinker()
        next(wifi_blinker)
        if not self.is_connected:
            self.logger.info("Try to connect to {} access point".format(self.ssid
))
            self.wlan.connect(self.ssid, self.password)

            timer = Timer(timeout=10,
                          tick_time=0.2)
            while timer.tick:
                if not self.is_connected:
                    wifi_blinker.send(self.is_connected)
                else:
                    self.logger.info("Successfully connected to {}".format(self.s
sid))
                    led_blink(blink_time=5)
                    break
            else:
                if self.status != 5:
                    if self.status == 2:
                        self.logger.error('Failed ot connect to {}. Wrong passwor
d'.format(self.ssid))
                    elif self.status == 3:
                        self.logger.error('Failed to connect to {}. No such AP'.f
ormat(self.ssid))
                else:
                    self.logger.error('Failed to connect to {}'.format(self.ssid)
)
        else:
            self.logger.info("Connected to {}".format(self.ssid))
```

Файл головного додатку

У класі App з-за допомогою усіх інших керуючих класів будується управління пристроєм. Реалізація знаходиться у методі start.

```
    def start(self):
        self.wifi.connect()
        while True:
            self.dht.measure()
            self.http_client.send_sensors_info(temperature=self.dht.temperature,
                                               humidity=self.dht.humidity)
            time.sleep(10)
```

ВИСНОВОК ДО РОЗДІЛУ 3

В даному розділі було створено програму для керування створеним прототипом на мові MicroPython. Програма виконує підключення до мережі, опит датчика температури та вологості та подальше відправлення даних на сервер.

Для полегшення користувачам налаштування пристрою, також був реалізований скрипт на мові Bash, що виконую основі функції (прошивка пристрою, пересилання програмних файлів).

РОЗДІЛ 4

НАЛАГОДЖЕННЯ ТА ПРИКЛАД РОБОТИ ПРИСТРОЮ

Встановлення ключових параметрів пристрою відбувається через файл settings.py, що представляє собою файл мови Python з константними змінними, що відповідають за той чи інший параметр налаштування. Ці параметри наступні:

- DEVICE_ID – унікальний ідентифікатор пристрою
- SERVER_ADDRESS – адреса серверу для надіслання даних
- SERVER_PORT – порт серверу
- SSID – назва безпровідної точки
- PASSWORD – пароль для доступу до Wi-Fi мережі

Наступні параметри використовуються для тестування пристрою в разі несправностей:

- LED_PIN – номер піну підключення світлодіоду до плати
- DHT_PIN – номер піну підключення датчика до плати
- GLOBAL_LOG_LEVEL – рівень логування у файл
- TO_STDOUT – прапорець, що вказує чи виводити налагоджувальної інформації в стандартний потік

Для полегшення взаємодії з девайсом, а саме прошивки контролеру, пересилання файлів, встановлення необхідних залежностей, був створений скрипт на мові Bash, реалізований у вигляді командного інтерфейсу виконує описані вище функції. Нижче представлений вихідний код скрипту.

```
#!/bin/bash

MICROPYTHON_BIN=esp8266-20191220-v1.12.bin
SRC_DIR=src/
```

```
DEVICE_PATH=/dev/ttyUSB0
PICOCOM_BAUDRATE=115200
function setup() {
    IS_INSTALLED=$(python3 -m pip freeze | grep esptool)
    if [ "$IS_INSTALLED" == "" ]; then
          echo "Installing esptool python package"
          pip3 install esptool > /dev/null || exit 1
    else
          echo "esptool package already installed"
    fi
    IS_INSTALLED=$(dpkg -l | grep picocom)
    # TODO: remove warning
    if [ "$IS_INSTALLED" == "" ]; then
          echo "Installing picocom"
          apt-get install -y picocom > /dev/null || exit 1
    else
          echo "picocom already installed"
    fi
}

function flash() {
    echo "Loading the firmare"
    esptool.py --port $DEVICE_PATH --baud 460800 write_flash --
flash_size=detect 0 $MICROPYTHON_BIN || exit 1
}

function load_files() {
    ampy --port $DEVICE_PATH put main.py || exit 1
    ampy --port $DEVICE_PATH put boot.py || exit 1
    ampy --port $DEVICE_PATH put $SRC_DIR || exit 1
}

function run_script() {
    echo "Running $1"
    ampy --port $DEVICE_PATH run "$1" || exit 1
}

function erase() {
    echo "Erasing the flash"
    esptool.py --port $DEVICE_PATH erase_flash || exit 1
}

function usage() {
    echo "Usage:"
    echo ""
    echo "    -h : script usage"
    echo "    -p : connect via picocom (n/a)"  # TODO: fix
    echo "    -i : install all required dependencies"
```

```
    echo "    -f : deploy the micropython firmware on the device"
    echo "    -l : load files with source code"
    echo "    -r : run python script and get the output"
    echo "    -e : erase the flash of the device"
    echo ""
}

while [ -n "$1" ]; do
    case $1 in
        -i)
            setup
            ;;
        -e)
            erase
            ;;
        -f)
            flash
            ;;
        -l)
            load_files
            ;;
        -r)
            shift
            run_script "$1"
            ;;
        -h | *)
            usage
            ;;
    esac
    shift
done
```

Для перевірки роботи створеного пристрою було створено простий HTTP сервер, який при отримані даних виводить їх у стандартний потік. Нижче представлений вихідний код тестового серверу. На рис. 4.1 зображений приклад роботи тестового серверу.

```
from http.server import BaseHTTPRequestHandler, HTTPServer

class HandleRequests(BaseHTTPRequestHandler):
    def _set_headers(self):
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()
```

```
    def do_GET(self):
        self._set_headers()
        self.wfile.write("received get request"

    def do_POST(self):
        self._set_headers()
        content_len = int(self.headers.get('content-length', 0))
        post_body = self.rfile.read(content_len)
        self.wfile.write("received post request:<br>{}".format(post_body).
encode())

    def do_PUT(self):
        self.do_POST()


if __name__ == '__main__':
    host = ''
    port = 8088
    HTTPServer((host, port), HandleRequests).serve_forever()
```

```
dima@hp:~/Desktop$ sudo python3 simple_http_server.py
127.0.0.1 - - [04/Jun/2020 13:45:49] "POST /?temp=24.3&hum=14 HTTP/1.1" 200 -
127.0.0.1 - - [04/Jun/2020 13:45:50] "POST /?temp=24.3&hum=14 HTTP/1.1" 200 -
127.0.0.1 - - [04/Jun/2020 13:45:56] "POST /?temp=24.4&hum=17 HTTP/1.1" 200 -
127.0.0.1 - - [04/Jun/2020 13:46:07] "POST /?temp=24.3&hum=22 HTTP/1.1" 200 -
127.0.0.1 - - [04/Jun/2020 13:46:09] "POST /?temp=24.3&hum=22 HTTP/1.1" 200 -
```

Рис. 4.1. Приклад роботи тестового серверу.

ВИСНОВОК ДО РОЗДІЛУ 4

В цьому розділі представлено інструкцію та скрипт для взаємодії користувача з створеним пристроєм. Описано наявні налаштування прошивки.

Для тестування працездатності девайсу було створено тестовий HTTP сервер. Що при отриманні HTTP запиту, виводить його тіло, де й знаходяться виміряні показання температури та вологості отримані від сенсору. Проблем під час роботи не виникало, та всі запити були успішно надіслані.

ВИСНОВКИ

В даній бакалаврській роботі був розроблений та реалізований у вигляді робочого прототипу, пристрій для моніторингу температури та вологості для системи «розумного дому». Був проведений аналіз існуючих рішень на ринку, їх переваги та недоліки та серед діапазону апаратного забезпечення в певних категоріях були обрані найбільш підходящі в даному випадку, виходячи с багатьох оціночних критеріїв (в тому числі економічного та естетичного).

В результаті досліджень мікроконтролерною системою було обрано плату NodeMCU v3 на базі системи на кристалі ESP8266. В ролі датчику температури та вологості – сенсор DHT21. Для індикації роботи пристрою та відображення зібраних даних (не є обов'язковим), використано світлодіод та OLED дисплей SSD1306. Після побудови та тестування прототипу на макетному стенді, компоненти були спаяні та поміщені в корпус.

З використанням програмної платформи MicroPython реалізовано програму керування пристроєм. При старті пристрою виконується його підключення до мережі. Після успішного під'єднання, проводиться вимірювання температури та вологості з датчика. Далі, отримані дані на сервер відсилаються з-за допомогою безпровідного Wi-Fi підключення по протоколу HTTP. Час опитування датчиків може бути сконфігурований. Створений допоміжний скрипт на мові Bash для полегшення прошивки та налаштування девайсу.

Створений виріб є цілком працездатним та готовим для використання у подібних системах моніторингу. Даний прототип має базовий набір функцій для комфорного використання по найнижчій ціні для виготовлення. В майбутньому, функціонал та можливості можна розширити шляхом додавання нових сенсорів, переходом на більш ефективні контролери або внесення у схему зовнішього джерела живлення.

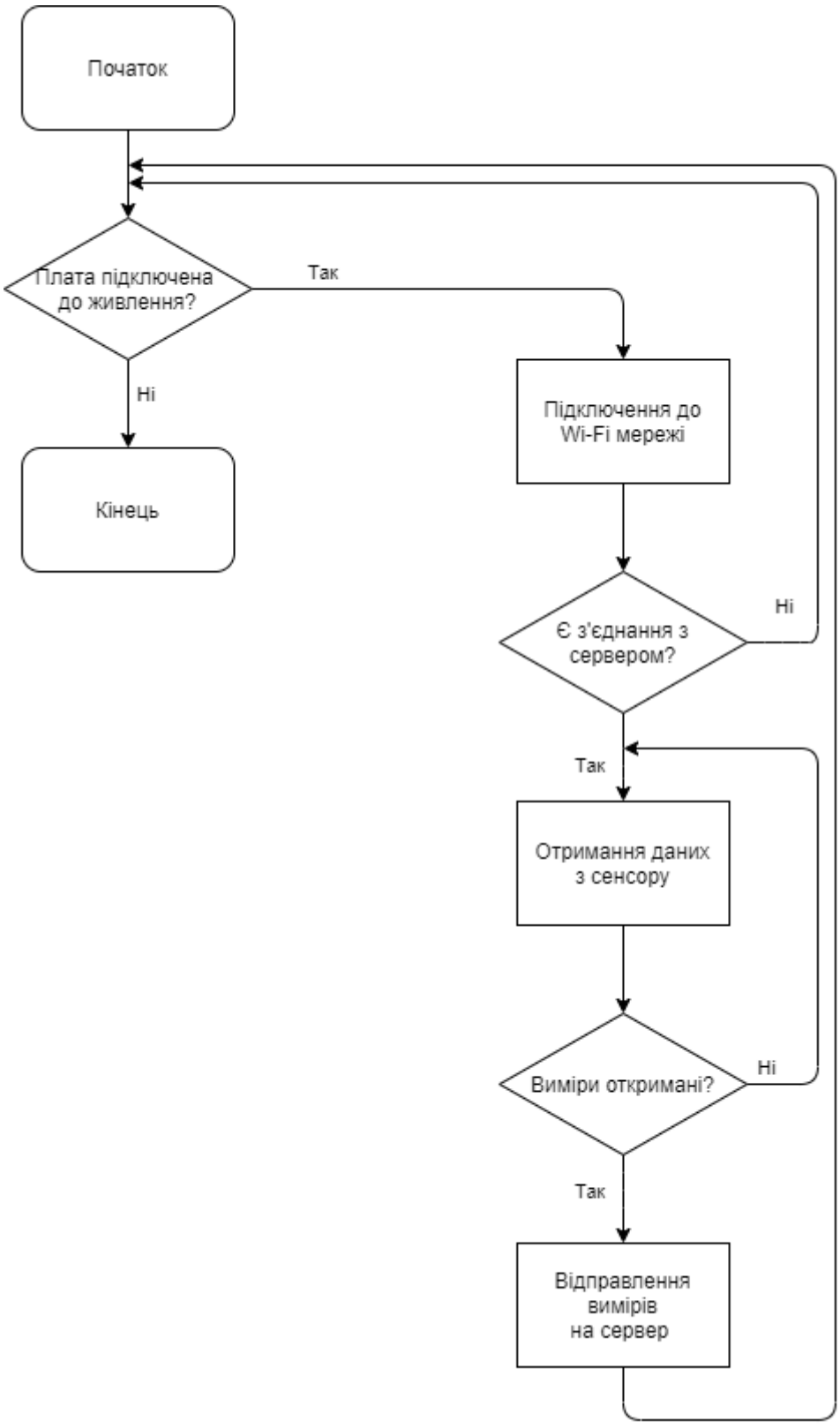
Цілком можна сказати, що ціль дипломної роботи була виконана. Досягнутий результат відповідає меті, а також має достатньо великий потенціал для масштабованості у багатьох напрямках. Обрана галузь роботи є перспективною у сучасному світі, тому є доцільним подальше удосконалення знань та робота в даному напряму.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

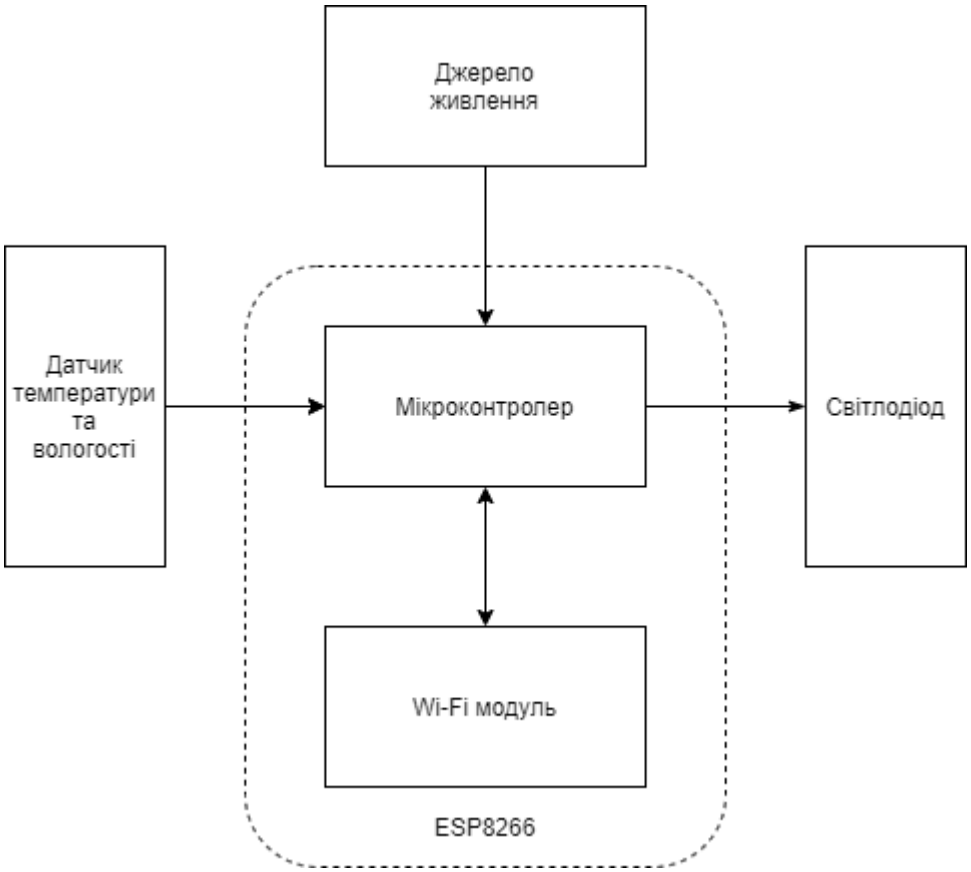
1. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
2. https://embeddedbharath.com/projects.html
3. https://www.raspberrypi.org/products/raspberry-pi-zero-w/
4. http://acoptex.com/project/8003/raspberry-basics-project-29a-raspberry-pi-zero-w-board-raspberry-pi-gpio-pinout-at-acoptexcom/
5. https://store.arduino.cc/arduino-ethernet-rev3-without-poe
6. https://store.arduino.cc/arduino-uno-rev3
7. https://store.arduino.cc/arduino-mkr-wifi-1010
8. https://store.arduino.cc/arduino-uno-wifi-rev2
9. https://www.espressif.com/en/products/socs
10. https://www.itransition.com/blog/iot-history
11. https://micropython.org
12. https://esphome.io/devices/nodemcu_esp32.html
13. https://www.electronicwings.com/sensors-modules/
14. https://randomnerdtutorials.com/guide-for-oled-display-with-arduino/
15. https://www.inspectorgadgets.ru/post/smart-home-explaine
16. Cuno Pfister. Getting Started with the Internet of Things; Make Community LLC, 2011
17. Maciej Kranz. Building the Internet of Things; Wiley, 2016

ДОДАТОК А

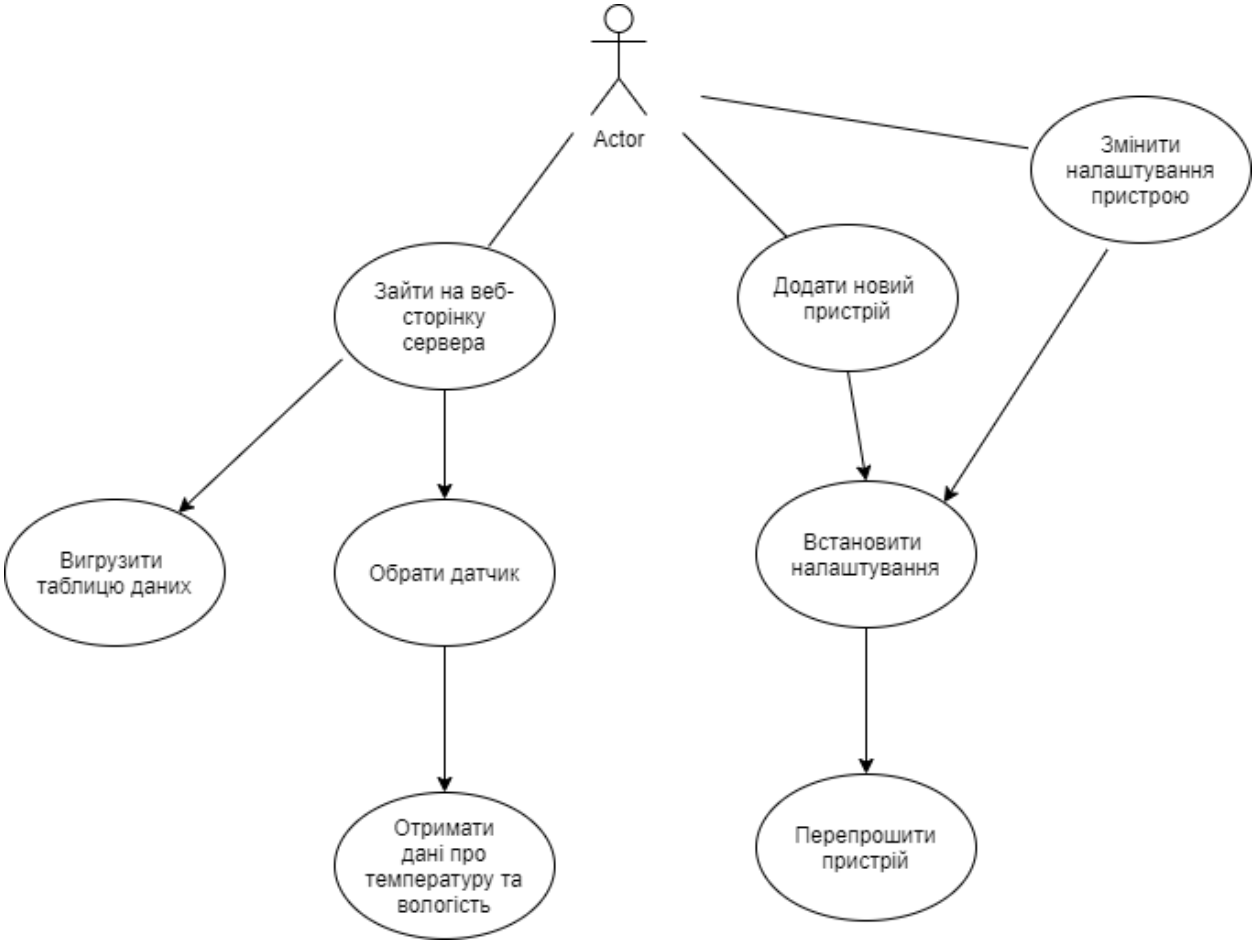
Київ – 2020



					ІАЛЦ.467100.004 Д1		
Зм.		*№ документа*	*Підп.*	*Дата*			
Розроб.		*Добровольський Д.Д.*			*Система моніторингу для розумного дому (клієнтська частина) Блок-схема роботи алгоритму пристрою*	*Літ.*	*Аркуш*
Перевір.		*Сімоненко В.П.*				Т	1 / 60
Н.конт		*Сімоненко В. П.*				*НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІО-61*	
Затв.							



					ІАЛЦ.467100.005 Д2		
Зм.		№ документа	Підп.	Дата			
Розроб.		Добровольський Д.Д.			Система моніторингу для розумного дому (клієнтська частина)	Літ.	Аркуш	
Перевір.		Сімоненко В.П.				Т	1	60
Н.конт		Сімоненко В. П.			Функціональна схема пристрою	НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІО-61	
Затв.							



					ІАЛЦ.467100.006 Д3		
Зм.		*№ документа*	*Підп.*	*Дата*			
Розроб.		*Добровольський Д.Д.*			*Система моніторингу для розумного дому (клієнтська частина)* *Структурна схема сценарію використання системи*	*Літ.*	*Аркуш*
Перевір.		*Сімоненко В.П.*				Т	1 / 60
Н.конт		*Сімоненко В. П.*				*НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІО-61*	
Затв.							